UNIVERSIDADE FEDERAL DO PARANÁ

CAROLINA DE ALMEIDA SANTOS PINOTTI

DESAFIOS NA APLICAÇÃO DE *PARTICLE SWARM OPTIMIZATION* EM UM PROBLEMA DE PLANEJAMENTO DE PRODUÇÃO DE UMA OLARIA

CURITIBA

2017

CAROLINA DE ALMEIDA SANTOS PINOTTI

DESAFIOS NA APLICAÇÃO DE *PARTICLE SWARM OPTIMIZATION* EM UM PROBLEMA DE PLANEJAMENTO DE PRODUÇÃO DE UMA OLARIA

Dissertação apresentada como requisito parcial à obtenção do Título de Mestre pelo Programa de Métodos Numéricos em Engenharia, no Programa de Pós Graduação em Métodos Numéricos em Engenharia, Setor de Tecnologia da Universidade Federal do Paraná
Orientadora: Profª Dra. Neida Maria Patias Volpi

CURITIBA

2017

P657d Pinotti, Carolina de Almeida Santos
Desafios na aplicação de particle swarm optimization em um problema de planejamento de produção de uma olaria / Carolina de Almeida Santos Pinotti. – Curitiba, 2017.
102 f. : il. color. ; 30 cm.

Dissertação - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Métodos Numéricos em Engenharia, 2017.

Orientadora: Neida Maria Patias Volpi.

1. Planejamento de produção. 2. Olaria. 3. Branch-andBound. 4. Métodos numéricos. I. Universidade Federal do Paraná. II. Volpi, Neida Maria Patias. III. Título.

CDD: 518

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
Setor TECNOLOGIA
Programa de Pós-Graduação MÉTODOS NUMÉRICOS EM ENGENHARIA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em MÉTODOS NUMÉRICOS EM ENGENHARIA da Universidade Federal do Paraná foram convocados para realizar a arguição da dissertação de Mestrado de **CAROLINA DE ALMEIDA SANTOS PINOTTI** intitulada: **DESAFIOS NA APLICAÇÃO DE PARTICLE SWARM OPTIMIZATION EM UM PROBLEMA DE PLANEJAMENTO DE PRODUÇÃO DE UMA OLARIA**, após terem inquirido a aluna e realizado a avaliação do trabalho, são de parecer pela sua aprovada.

Curitiba, 03 de Março de 2017.

NEIDA MARIA PATIAS VOLPI
Presidente da Banca Examinadora (UFPR)

LUIZ CARLOS DE ABREU RODRIGUES
Avaliador Externo (UTFPR)

VOLMIR EUGÊNIO WILHELM
Avaliador Interno (UFPR)

Dedico este trabalho à minha família,
que sempre me apoiou em minhas decisões,
mesmo as mais loucas.

AGRADECIMENTOS

Agradeço à CAPES por financiar meus estudos junto à Universidade Federal do Paraná. Agradeço também ao Programa de Pós-Graduação em Métodos Numéricos em Engenharia (PPGMNE) da Universidade Federal do Paraná pela oportunidade de me tornar Mestre por intermédio do programa.

À minha orientadora Neida Maria Patias Volpi pela orientação e paciência, principalmente quando tive dificuldades em focar na pesquisa. Agradeço muito também por confiar em mim, aceitando me orientar e acreditando que eu conseguiria realizar o trabalho no meio de tantos problemas.

Ao meu amigo Renato, por permitir que eu fizesse a pesquisa baseada em sua olaria, pela atenção e paciência em me explicar o processo e por me deixar livre para conhecer o local e pegar todos os dados que precisasse. Sua ajuda foi fundamental para o desenvolvimento dessa pesquisa.

Ao professores Arinei Carlos Lindbeck da Silva, Gustavo Valentim Loch e Paulo Henrique Siqueira e aos colegas Joyce, Paulo e Sander por contribuírem com alguns detalhes da programação em Visual Basic, Matlab e Lingo. Ao Jair por todo o suporte e ajuda durante o curso. A todos os professores que, de forma direta ou indireta, participaram de minha formação.

Aos meus amigos Matheus e Bruno de Lessa por me incentivarem a fazer o mestrado e não me deixarem desanimar ou desistir. Às minhas amigas Dayane, Thamara e Jackeline, pelas diversas horas de estudo e aprendizado durante esses três anos, aliadas também a muitas conversas e risadas. Ao Claudio pela indispensável ajuda com o Visual Basic e Lingo, pelas horas e horas de trabalho em cima do programa.

Agradeço em especial aos meus pais, Regina e Celso, e meus irmãos, Nícolas e Melissa, pelo apoio durante todo o tempo, pela estrutura familiar que me possibilitou condições para fazer o mestrado e por todo o amor que me dá forças para tudo.

Aos meus amigos Rute, Mônica, Aline, Thiago Sato e Bruna, por sempre estarem do meu lado, acreditando em mim e me dando forças para superar todos os problemas. À Shara, Kika, Marise, Sálua, Liu, Bia, Xeda e Sandra por todas as noites, eventos e coreografias juntas em prol de um amor em comum que é a dança.

"[...] A luta pela vida nem sempre é vantajosa aos fortes nem aos espertos. Mais cedo ou mais tarde, quem cativa a vitória é aquele que crê plenamente EU CONSEGUIREI!"

(Napoleon Hill, Filosofia do Sucesso)

RESUMO

O presente trabalho tem como objetivo principal verificar a aplicabilidade de uma *math*-heurística em um problema de Planejamento de Produção de uma indústria de tijolos. Foi escolhida a metaheurística *Particle Swarm Optimization* (PSO) para a aplicação no problema. Depois de realizada uma revisão bibliográfica acerca do assunto percebeu-se que são encontrados poucos trabalhos que aplicam um PSO discreto em um problema de planejamento de produção, o que motivou o trabalho. A *math*-heurística será aqui a combinação do PSO com a metodologia exata *Branch-and-Bound*. A combinação deve-se ao fato de que a aplicação proposta é feita com um PSO discreto, ou seja, a matriz possui somente entradas inteiras. O uso do PSO discreto fez com que fosse proposta a utilização de uma metodologia exata para que a função objetivo do problema fosse encontrada, sendo que esta depende da distribuição das quantidades a serem produzidas e esses valores, por sua vez, são encontrados com base na matriz discreta que está sendo movimentada pelo PSO. O problema proposto possui preservação da preparação, capacidade (tempo limite de produção) e demanda a ser atendida. A preservação da preparação é o que dificulta a resolução do problema por heurísticas (como o *Greedy-Mod*) e também faz com que se encontrem muitas infactibilidades. Os desafios encontrados na aplicação dessa proposta não foram poucos. Merece uma maior discussão a codificação e representação das soluções, tratamento de infactibilidades nas soluções iniciais e na movimentação da nuvem. Estes problemas geraram alto custo computacional. Sugere-se uma tentativa de mudança na forma de aplicação (para a utilização de um PSO contínuo), de forma que as infactibilidades sejam reduzidas e o tempo computacional melhorado.

Palavras-chaves: PSO. Planejamento de Produção. Olaria. Metodologia exata. *Branch-and-Bound*.

ABSTRACT

The present work has as main objective to verify the applicability of a math-heuristic in the Production Planning of a brick industry. The Particle Swarm Optimization (PSO) metaheuristic was chosen for the application in the problem. After a bibliographical review about the subject, it was noticed that not too many works were found that apply a discrete PSO in a problem of production planning, one of the reasons that motivated this work. The math-heuristic will be the combination of the metaheuristic PSO with the Branch-and-Bound exact methodology (performed by an optimizer). The combination is due to the fact that the proposed application is made with a discrete PSO, that is, the matrix to be moved has only discrete inputs. The use of discrete PSO has meant that the use of an exact methodology is proposed so that the objective function of the problem is found, which depends on the distribution of the quantities to be produced and these values, in turn, are found based on the discrete matrix that is being moved by the PSO. The proposed problem has preservation of the preparation, capacity (time limit of production) and demand to be met. Preservation of the preparation is what makes it difficult to solve the problem by heuristics (such as Greedy-Mod) and also causes many infeasibilities. Deserves a further discussion the codification and representation of solutions, treatment of infeasibilities in the initial solutions and in the movement of the swarm. These problems generated a high computational cost. An attempt is made to change the manner of application (for the use of a continuous PSO), so that the infeasibilities are reduced and the computational time improved.

Key-words: PSO. Production Planning. Brickyard. Exact methodology. Branch-and-Bound.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo serão abordados a motivação do trabalho desenvolvido, objetivos geral e específicos e uma breve apresentação dos capítulos.

1.1 MOTIVAÇÃO DO TRABALHO

O número de indústrias instaladas no Brasil e no mundo é muito grande. As indústrias de grande porte geralmente possuem um gerente de produção, que controla o que é produzido e estocado pela indústria em um determinado período de tempo.

Uma grande dificuldade encontrada pelas indústrias é quanto à melhor forma de produzir os itens para gerar um menor custo ou também gerar um maior lucro. Existem vários problemas de otimização encontrados em uma linha de produção, e cada indústria possui pelo menos um. Alguns dos que podem ser matematicamente estudados são quanto ao planejamento, dimensionamento, sequenciamento e programação da produção.

Os problemas de pequeno porte podem ser abordados, matematicamente, de diversas formas. Com o uso de métodos exatos da pesquisa operacional em problemas de pequeno porte a solução ótima é facilmente encontrada, sendo estes resolvidos manualmente ou também com a ajuda de *softwares* otimizadores, que são na maioria das vezes muito eficientes. Porém, conforme o problema aumenta de complexidade, tais metodologias de resolução tornam-se muitas vezes desvantajosas, com um custo computacional muito elevado. Assim, faz-se necessário o uso de outros métodos, como é o caso das heurísticas e metaheurísticas, que são utilizados com o intuito de resolver problemas de modelagem matemática obtendo um menor custo. Fazendo uma comparação, é possível definir vantagens e desvantagens de cada metodologia de resolução, diferença de custo de implementação e, assim, aplicar esses outros tipos de resolução em problemas de grande porte, dos quais não é possível determinar a solução ótima por métodos exatos.

Como objetivo geral deste trabalho tem-se a resolução de um problema de produção por um método matemático. Existem muitas aplicações de heurísticas e metaheurísticas em problemas de produção, mas não específicas de PSO no problema de planejamento. São comumente encontradas aplicações de *Particle Swarm Optimization* (PSO), Algoritmo Genético (GA - *Genetic Algorithm*), Busca Tabu e outras metaheurísticas em problemas de *Flow Shop* ou *Job Shop*, que são problemas de sequenciamento de produção em ambiente de máquina. Em problemas de Planejamento de Produção são encontrados poucas aplicações de PSO discreto, o que motivou este trabalho.

Este trabalho realizará, portanto, a aplicação da metaheurística PSO e de uma metodologia exata em um problema de Programação da Produção. Será analisada se tal

aplicação é vantajosa para casos em que os métodos exatos não encontram a solução ótima com um tempo computacional adequado. Nestes casos, a aplicação da metaheurística é geralmente de grande eficiência quando combinada com uma linguagem de programação.

Após a aplicação em um problema geral, ainda será aplicada essa metodologia em um problema de uma indústria de tijolos, como forma de exemplificar o problema com uma situação real. Para a eficiência da aplicação é preciso, primeiramente, construir o modelo matemático que descreve um problema de produção de uma olaria. Deve-se também estudar a metaheurística para saber aplicá-la adequadamente ao problema, fazendo as modificações necessárias. Por fim, é feita a aplicação da metaheurística, combinada com um método exato (*Math*-heurística), ao modelo construído e são comparados os resultados encontrados pela resolução combinada com os resultados encontrados com somente métodos exatos (nos casos onde isto é possível).

1.2 OBJETIVOS GERAL E ESPECÍFICOS

O objetivo geral deste trabalho é verificar a aplicabilidade da metaheurística PSO e uma metodologia exata em um problema de produção.

Como objetivos específicos:

- Estudar os problemas de produção com o fim de entender as diferenças e possíveis aplicações;
- Estudar o problema de planejamento de produção;
- Pesquisar as aplicações já existentes de metaheurísticas em problemas de produção;
- Pesquisar as aplicações de metaheurísticas no problema de planejamento de produção;
- Estudar a metaheurística PSO;
- Estudar a forma de aplicar a metaheurística PSO no problema de planejamento de produção em uma indústria de tijolos;
- Desenvolver um programa em linguagem computacional para a aplicação;
- Fazer testes computacionais em cima do modelo programado e analisar a conveniência do uso desta ferramenta.

1.3 APRESENTAÇÃO DOS CAPÍTULOS

O primeiro capítulo é este que trata da introdução, onde foram apresentados os objetivos gerais e específicos e a motivação do trabalho. A seguir encontra-se a revisão bibliográfica, que tratará dos diversos trabalhos encontrados na literatura acerca deste tema,

como metaheurísticas, PSO, problemas de produção e problema de planejamento de produção. Pode-se perceber a vasta aplicação de metaheurísticas em problemas de produção, mas uma não tão grande aplicação de PSO em planejamento de produção. Na revisão bibliográfica também será encontrado um estudo sobre o problema de planejamento de produção e a metaheurística PSO.

O terceiro capítulo mostrará a proposta do uso do PSO no problema escolhido. É feita uma adaptação do método para uma melhor aplicação ao problema, e é o que esse capítulo estará apresentando. Além disso, será abordado também sobre a programação computacional realizada, e de que forma foram feitos os testes de verificação para uma análise da aplicação.

O capítulo seguinte apresentará uma aplicação desta proposta em uma indústria de tijolos. Serão apresentados o problema, os dados da indústria, o modelo adaptado a essa indústria e os testes computacionais finais.

Por fim, tem-se a conclusão e as referências, seguidas dos Apêndices e Anexos, onde serão apresentados a programação computacional realizada e também imagens e/ou gráficos pertinentes.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo contém uma introdução dos problemas que serão tratados. Tem-se uma seção abordando os Problemas de Produção e outra a metaheurística utilizada, o PSO. Para uma maior familiaridade com o assunto, apresentou-se cada um na forma original para depois fazer as adaptações necessárias.

2.1 PROBLEMAS DE PRODUÇÃO

Os problemas de produção podem ser classificados em grupos tais como: planejamento da produção, dimensionamento de lotes e de programação da produção. Segundo (ARENALES et al., 2007), os problemas na área de produção são classificados em três níveis: estratégico, tático e operacional. O primeiro nível, o estratégico, trata da escolha e do projeto do processo, o nível tático acerca do planejamento das atividades e o operacional controla as atividades diárias, designando tarefas às máquinas e programando as tarefas em cada máquina.

Em se tratando de problemas de produção existe um campo muito amplo de pesquisas já realizadas e também a serem ainda desenvolvidas. Nos três níveis encontramos diferentes tipos de problemas, como citados anteriormente. Sobre programação da produção, os mais estudados são *Flow Shop* e *Job Shop*, que tratam de sequenciamento. O problema de planejamento de produção possui um horizonte de planejamento finito, é dividido em períodos e tem uma demanda dinâmica (que varia ao longo do horizonte). Este pode ser representado por um modelo determinístico, que é o modelo a ser tratado neste trabalho.

Vários parâmetros e dados são necessários e utilizados para a resolução do problema. Pode-se falar de demanda (dinâmica ou fixa), capacidade, custos de produção, custos de estoque, tempo de produção, entre outros. Tem-se também aqueles que consideram a preservação da preparação, ou seja, que a máquina permanece preparada para algum item de um período a outro. Tudo isso gera restrições para o problema que podem facilitar ou dificultar a procura de uma solução otimizada.

2.1.1 Problema de Planejamento de Produção

Os problemas de planejamento de produção são diversos, dependem das restrições encontradas no modelo. Podem conter um item ou múltiplos itens, capacidade, restrição de preservação da preparação, atraso, entre outros, dependendo muito do objetivo e também da aplicação escolhida.

Os modelos de planejamento de produção podem ser divididos em vários casos . Neste planejamento o dimensionamento dos lotes é variável. Aqui serão explanados alguns deles: “Um item sem restrição de capacidade”, “Múltiplos itens com restrição de capacidade” e “Múltiplos

itens com restrição de capacidade e preservação da preparação", todos definidos ao longo de um horizonte de planejamento (ARENALES et al., 2007).

Serão apresentados a seguir os três modelos citados anteriormente. Para cada item i e período t, definem-se:

Parâmetros:

N : número de itens do problema

T : número de períodos

Índices:

i : itens

t : períodos

Dados:

b_i : capacidade (em unidades de tempo - u.t.) necessária do recurso da máquina para produzir uma unidade do item i

C_t : capacidade total (em u.t.) disponível do recurso da máquina no período t

d_{it} : demanda do item i atendido ao final do período t

h_i : custo de estoque do item i por período

s_i : custo de *setup* do item i

sp_i : tempo de *setup* (em u.t.) da máquina para processar o item i

Condições Iniciais:

I_{i0} : estoque inicial do item i

Variáveis:

I_{it} : estoque do item i no final do período t

q_{it} : quantidade do item i a ser produzida no período t

y_{it} : $\begin{cases} 1 \text{ se o item } i \text{ é produzido no período } t \\ 0 \text{ caso contrário} \end{cases}$

O primeiro deles, com um item sem restrição de capacidade, é o modelo mais simples de dimensionamento de lotes. A função objetivo minimiza o custo total de preparação e estoque, as restrições limitam a produção de acordo com a demanda e fazem o balanceamento do estoque. A formulação do modelo é apresentada a seguir:

$$min \sum_{t=1}^{T} (s_1 y_{1t} + h_1 I_{1t}) \tag{2.1}$$

$$I_{1t} = I_{1,t-1} + q_{1t} - d_{1t} \qquad t = 1, \ldots, T \;\; e \;\; I_{10} = I_{1T} = 0 \tag{2.2}$$

$$q_{1t} \leq \left(\sum_{\tau=t}^{T} d_{1\tau} \right) \cdot y_{1t} \qquad t = 1, \ldots, T \tag{2.3}$$

$$q \in \mathbb{R}_{+}^{1T}, \; I \in \mathbb{R}_{+}^{1T}, \; y \in \mathbb{B}^{1T} \tag{2.4}$$

No segundo caso, de múltiplos itens com capacidade, além da exigência da demanda (por item) existe a limitação pela capacidade (por máquina). A máquina não pode produzir mais do que sua capacidade de produção. O valor de M_{it} limita superiormente a produção do item no período considerando a capacidade resultante e a demanda futura.

$$min \sum_{i=1}^{N} \sum_{t=1}^{T} s_i y_{it} + h_i I_{it} \tag{2.5}$$

$$I_{it} = I_{i,t-1} + q_{it} - d_{it} \qquad i = 1, \ldots, N \;\; e \;\; t = 1, \ldots, T \tag{2.6}$$

$$\sum_{i=1}^{N} sp_i y_{it} + b_i q_{it} \leq C_t \qquad t = 1, \ldots, T \tag{2.7}$$

$$q_{it} \leq M_{it} y_{it} \qquad i = 1, \ldots, N \;\; e \;\; t = 1, \ldots, T \tag{2.8}$$

$$M_{it} = min \left\{ \frac{C_t - sp_i}{b_i}, \sum_{\tau=t}^{T} d_{i\tau} \right\} \qquad i = 1, \ldots, N \;\; e \;\; t = 1, \ldots, T \tag{2.9}$$

$$q \in \mathbb{R}_{+}^{NT}, \; I \in \mathbb{R}_{+}^{NT}, \; y \in \mathbb{B}^{NT} \tag{2.10}$$

O terceiro, e último caso citado, cuja formulação será apresentada na seção a seguir, além de possuir múltiplos itens e restrição de capacidade, ainda possui a preservação da preparação. Esta última condição leva a máquina a ficar preparada para um item de um período o outro, ou seja, não há a necessidade de preparar a máquina caso o primeiro item a ser produzido neste período seja o mesmo item produzido no final do período anterior. Assim, o sequenciamento da produção é diferente e o custo é menor do que em um modelo que não considera a preservação da preparação.

Pode-se considerar para todos esses problemas a possibilidade de demanda entregue com atraso ou sem atraso. Caso o atraso seja possível, na função objetivo deve ser considerada a penalidade por demanda entregue atrasada.

Desses três modelos de planejamento de produção citados anteriormente, dos quais dois já foram apresentados, o que este trabalho utilizará é o modelo com múltiplos itens,

restrição de capacidade e preservação da preparação, cuja formulação será apresentada na seção a seguir.

2.1.2 Planejamento de produção com múltiplos itens, restrição de capacidade e preservação da preparação

Nesta seção será apresentado o problema de planejamento da produção com múltiplos itens, restrição de capacidade e preservação da preparação. É um modelo que possui uma solução ótima melhor do que a solução dos modelos sem a preservação da preparação, já que a máquina não precisa ser preparada todos os períodos (e isso inclusive será minimizado na função objetivo), mas também possui uma dificuldade maior na modelagem.

O problema a ser trabalhado e sua modelagem matemática encontram-se a seguir, e nos próximos capítulos será feita uma aplicação de um método de resolução matemático diferente do método exato para a resolução deste problema.

Considerando N o número de itens a serem produzidos em uma linha de produção durante T períodos, com i item, tal que, $i = 1, \ldots, N$, e t período, tal que $t = 1, \ldots, T$. Neste problema tem-se que cada item está sujeito a uma demanda, um custo de estoque por período e um tempo de produção. Cada período possui uma capacidade limite. Além disso, há um tempo de preparação da máquina (tempo de *setup*), um custo de preparação da máquina (custo de *setup*) e custo de estoque para um determinado item.

O problema a ser trabalhado possui a seguinte notação:

Parâmetros:

M : número grande

N : número de itens do problema

T : número de períodos

Índices:

i : itens

t : períodos

Dados:

b_i : capacidade (em unidades de tempo - u.t.) necessária do recurso da máquina para produzir uma unidade do item i

C_t : capacidade total (em u.t.) disponível do recurso da máquina no período t

d_{it} : demanda do item i atendido ao final do período t

h_i : custo de estoque do item i por período

s_i : custo de *setup* do item i

sp_i : tempo de *setup* (em u.t.) da máquina para processar o item i

Condições Iniciais:

I_{i0} : estoque inicial do item i

Variáveis:

I_{it} : estoque do item i no final do período t

q_{it} : quantidade do item i a ser produzida no período t

Algumas outras variáveis serão definidas para a formulação do problema. A variável y_{it} retornará o valor 1 quando a máquina for preparada (*setup*) para aquele item i no período t, e isto incorre em custo. Se houve produção do item i no período $t-1$ e a máquina ficou preparada para este item até o período t, diz-se que houve preservação da preparação. Então a variável y_{it} retornará 1 quando a máquina tiver sido preparada para o item i no período t ou 0 se teve a preparação mantida do período anterior a esse período. A variável w_{it} retorna o valor 1 quando houve preservação da preparação do item i no período t e a variável z_t retorna o valor 1 se algum item tiver sido produzido em um período $t-1$, no período t e a preparação ainda tiver sido mantida até o período $t+1$. Ou seja, no período t o item i foi o único a ser produzido (pois a preparação da máquina para este item foi mantida até o período $t+1$). Dessa forma, as variáveis serão definidas como a seguir.

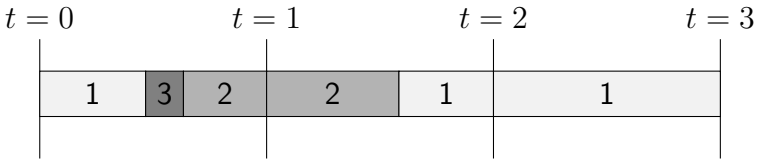
$$w_{it}: \begin{cases} 1, & \text{se o estado de preparação para o item } i \text{ continua do período } t-1 \text{ ao período } t \\ 0, & \text{caso contrário} \end{cases}$$

$$y_{it}: \begin{cases} 1, & \text{se ocorreu } \textit{setup} \text{ na máquina para o item i no período t} \\ 0, & \text{caso contrário} \end{cases}$$

$$z_t: \begin{cases} 1, & \text{se não existe preparação de itens em t, sendo que a preparação de um item específico ocorreu em t-1 e é mantida até o período t+1} \\ 0, & \text{caso contrário} \end{cases}$$

Para uma maior compreensão dessas últimas variáveis definidas, a Figura 1 simula uma sequência de produção, onde cada cor representa um item a ser produzido. A seguir estarão exemplificadas as variáveis y, w e z, e seus respectivos valores para cada período dessa sequência exemplificada.

Figura 1 – SEQUÊNCIA DE PRODUÇÃO



Na Figura 1 tem-se uma linha de produção com três itens. No primeiro período, $t = 1$,

a produção foi na seguinte sequência: item 1 (cor de tonalidade mais clara), item 3 (cor mais escura) e item 2 (cor média). O item 2 teve sua preparação mantida do período 1 ao período 2, ou seja, $y_{22} = 0$ e $w_{22} = 1$. O item 1 não teve sua preparação mantida mas teve produção no período 2, logo, $y_{12} = 1$ e $w_{12} = 0$. Como teve preparação de um item neste período (o item 1), $z_2 = 0$. No período 3, observa-se que o item 1 teve sua preparação mantida, portanto, $y_{13} = 0$, $w_{13} = 1$. Os demais itens não tiveram produção, ou seja, $y_{23} = 0$ e $y_{33} = 0$. Assim, como não houve preparação de nenhum item neste período, $z_3 = 1$.

Com os dados, parâmetros e variáveis todos definidos, pode ser apresentada a formulação do modelo proposto:

$$min \sum_{i=1}^{N} \sum_{t=1}^{T} (s_i y_{it} + h_i I_{it}) \tag{2.11}$$

$$I_{it} = I_{i,t-1} + q_{it} - d_{it} \qquad i = 1, \ldots, N \;\; e \;\; t = 1, \ldots, T \tag{2.12}$$

$$\sum_{i=1}^{N} (sp_i y_{it} + b_i q_{it}) \leq C_t \qquad t = 1, \ldots, T \tag{2.13}$$

$$\sum_{i=1}^{N} w_{it} \leq 1 \qquad t = 1, \ldots, T \tag{2.14}$$

$$w_{it} \leq y_{i,t-1} + w_{i,t-1} \qquad i = 1, \ldots, N \;\; e \;\; t = 2, \ldots, T \tag{2.15}$$

$$w_{i,t+1} + w_{it} \leq 1 + z_t \qquad i = 1, \ldots, N \;\; e \;\; t = 1, \ldots, T-1 \tag{2.16}$$

$$y_{it} + z_t \leq 1 \qquad i = 1, \ldots, N \;\; e \;\; t = 1, \ldots, T \tag{2.17}$$

$$q_{it} \leq M \, (y_{it} + w_{it}) \qquad i = 1, \ldots, N \;\; e \;\; t = 1, \ldots, T \tag{2.18}$$

$$z \in \mathbb{R}_{+}^{T},\ q \in \mathbb{R}_{+}^{NT},\ I \in \mathbb{R}_{+}^{NT},\ y \in B^{NT} \text{ e } w \in B^{NT} \tag{2.19}$$

Com relação ao modelo, a função objetivo (equação 2.11) minimiza a soma dos custos de *setup* da máquina e custos de estoque para todos os itens i em todos os períodos t. As equações 2.12 correspondem ao balanceamento do estoque. As restrições 2.13 limitam produção à capacidade do período considerando os tempos de *setup* da máquina para todos os itens e tempo de produção de cada item de acordo com a quantidade de itens a ser produzida no período. As restrições 2.14 garantem que a máquina vai manter o estado de preparação para, no máximo, um item em cada período. As restrições 2.15, por sua vez, garantem que se não houve *setup* do item i e também não houve manutenção de preparação do item i no

período anterior, então também não poderá manter a preparação para este mesmo item no período atual. As restrições 2.16 garantem que se há preparação de algum item no período t ($z_t = 0$) não pode haver um item i que tem sua preparação preservada de $t-1$ a t e também de t a $t+1$. As restrições 2.17 garantem que somente uma das duas coisas pode acontecer: preparação da máquina para o item i no período t ou preservação da preparação de itens do período $t-1$ ao período $t+1$. As restrições 2.18 garantem que se não houve preparação da máquina para o item i no período e também o estado de preparação do item i não foi preservado do período $t-1$ para o período t, então não pode haver produção naquele período para aquele item i. Por último, as restrições 2.19 indicam o tipo das variáveis.

2.2 METAHEURÍSTICAS

O método matemático estudado para a aplicação no problema de produção foi uma metaheurística, que será apresentada a seguir, desde seu surgimento até o algoritmo utilizado.

2.2.1 Metodologia de Solução - PSO

A técnica de otimização PSO é uma metaheurística criada há mais de 20 anos por James Kennedy e Russell Eberhart, sendo apresentada pelos dois autores no artigo KENNEDY; EBERHART (1995). A seguir será dada uma introdução à ideia dessa metaheurística.

O PSO surgiu de vários estudos do comportamento de peixes e pássaros. Duas coisas que foram notadas nesses estudos é que os pássaros mantêm uma distância ótima entre eles e seus vizinhos. Os peixes, como membros individuais do bando, podem otimizar sua busca por alimentos baseando-se em experiências anteriores e descobertas dos outros membros do grupo. Mas pássaros e peixes adaptam seus movimentos físicos para otimizar a busca por comida. Os seres humanos ajustam não somente os movimentos físicos mas também utilizam o movimento cognitivo e as experiências. O estudo inicial dessa metaheurística, portanto, também é importante para desenvolver uma simulação mais fiel do comportamento humano.

Baseando-se na individualidade e sociabilidade, cada partícula possui um peso para se espelhar em suas melhores posições e também nas melhores posições do grupo inteiro, fazendo com que a otimização seja mais precisa e não caia diretamente em mínimos locais como aconteceria se fosse baseada somente na sociabilidade. A individualidade gera uma diversificação mas, também, quando incluída sozinha no modelo, tem-se várias partículas isoladas no espaço problema. Tais pesos serão incluídos no problema na forma de parâmetros, denominados aqui de η_1 e η_2.

Na parte da individualidade tem-se a variável chamada de *pbest*, que funciona como uma memória autobiográfica, e a velocidade associada à essa variável ajuda a partícula a retornar ao lugar que esteve mais satisfeito no passado. Por outro lado, *gbest* é a variável relacionada à sociabilidade, funcionando como um conhecimento público, com uma velocidade

que leva ao melhor lugar que uma partícula entre todas do grupo já esteve.

A velocidade de uma partícula é dada por três fatores: termo de inércia, termo cognitivo e termo de aprendizado social. Ao atualizar uma partícula deve-se avaliar se a solução encontrada é melhor ou pior que a anterior (melhor solução da partícula é chamada de *pbest*) e também se é melhor ou pior que a melhor solução encontrada até o momento (melhor solução de todas as partículas é chamada de *gbest*). Cada termo da velocidade da partícula determina um fator na equação da velocidade. O primeiro termo é dado pela inércia (parâmetro w com a velocidade anterior), o segundo é o termo cognitivo (*pbest* com o parâmetro η_1 e um valor aleatório) e o terceiro é o termo de aprendizado social (*gbest* com o parâmetro η_2 e um valor aleatório).

Seja x a posição atual e v a velocidade atual da partícula P. Após a movimentação, a nova posição será representada por x' e a nova velocidade por v'. A atualização da velocidade pode ser feita pela equação a seguir:

$$v_i' = w \cdot v_i + \eta_1 \cdot rand_1 \cdot (pbest_i - x_i) + \eta_2 \cdot rand_2 \cdot (gbest_i - x_i) \tag{2.20}$$

Com a atualização da velocidade pode-se atualizar a posição, da forma:

$$x_i' = x_i + v_i' \tag{2.21}$$

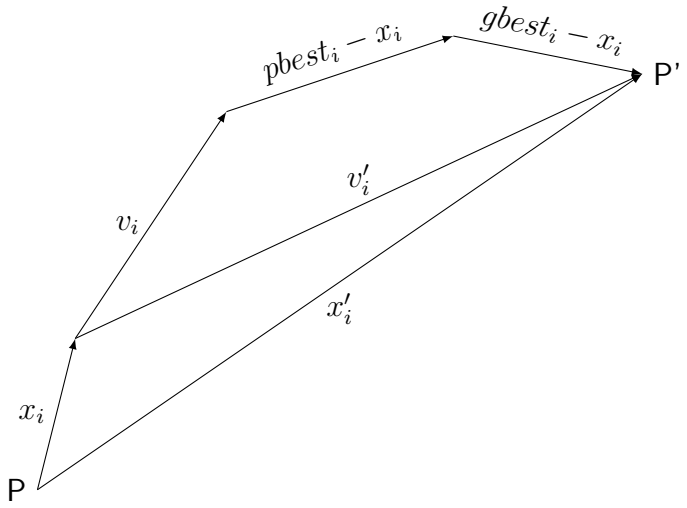
Sendo que:

- η_1 e η_2 são parâmetros do termo cognitivo e termo de aprendizado social, respectivamente;
- $rand_1$ e $rand_2$ são valores aleatórios entre 0 e 1;
- $pbest_i$ é a melhor posição que a partícula i já esteve;
- $gbest_i$ é a melhor posição que algum vizinho de i já esteve;
- w é o peso da inércia.

A movimentação de uma partícula da nuvem de P para P' pode ser visualizada na Figura 2 (supondo $w = 1$, $\eta_1 \cdot rand_1 = 1$ e $\eta_2 \cdot rand_2 = 1$).

MARINI; WALCZAK (2015) fazem uma revisão sobre a metaheurística, inserindo temas mais específicos do PSO, como escolha dos parâmetros, limitação da velocidade, inicialização da posição e velocidade e ainda faz uma abordagem sobre o PSO discreto. Com alguns exemplos é possível visualizar graficamente a aplicação e movimentação do PSO.

Figura 2 – VETOR x'_i



2.2.2 Codificação

A metaheurística PSO baseia-se na movimentação de uma Nuvem de Partículas, ou seja, são necessárias vários indivíduos na população para que, quando movimentados, melhorem a solução ótima do problema, levando em consideração o termo de inércia, cognitivo e social. A avaliação de cada um individualmente é importante para que a nuvem seja direcionada para a solução ótima global.

A movimentação de cada indivíduo do PSO gera um valor *fitness*, que é uma avaliação da posição onde este se encontra. Este valor pode ser calculado com base na função objetivo do modelo, para avaliar se o indivíduo está se aproximando ou se distanciando do valor ótimo do problema.

Os indivíduos da população inicial são atualizados na primeira iteração e movimentados, para um lugar melhor ou pior, dependendo da velocidade e direção para a qual estão indo. Caso haja melhoria em algum indivíduo, o *pbest* é atualizado e, caso essa seja a melhor solução encontrada, o *gbest* é atualizado também. Na sequência serão movimentados esses indivíduos obtidos na primeira iteração. Da mesma forma, independente do resultado da movimentação, os novos indivíduos são os encontrados nessa iteração e a atualização do *pbest* e *gbest* é realizada quando necessária. O mesmo será feito até a última iteração (quando atingir um critério de parada).

O critério de parada pode ser determinado por uma combinação do número de iterações, tempo computacional e também por uma avaliação das soluções que estão sendo encontradas (ou seja, caso os indivíduos já estejam encontrando uma mesma solução há um pré-determinado número de iterações, o critério de parada é atingido pois a nuvem encontra-se em um mínimo ou máximo local ou global).

O programa é finalizado quando o critério de parada é atingido. O melhor *fitness*

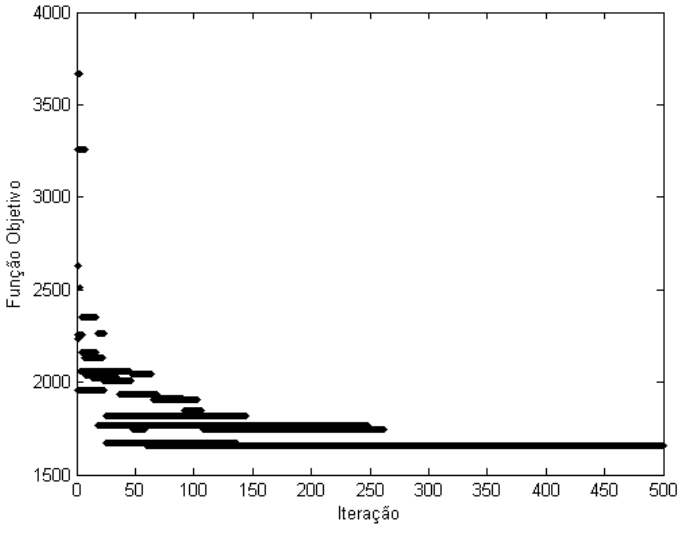
encontrado até o momento será o melhor valor para a função objetivo do modelo programado e a solução que gera esse *fitness* é a melhor solução encontrada, podendo esta ser a solução ótima ou não.

Resumindo o algoritmo PSO geral:

- Gera uma população inicial;
- Movimenta população com o PSO (atualiza velocidade e posição da partícula P);
- Atualiza o *pbest*, *gbest* e *fitness*;
- Verifica se satisfaz o critério de parada.

A Figura 3 mostra a movimentação de uma nuvem de partículas, sendo o eixo x representado pelas iterações (total de 300) e o eixo y pelo valor da função objetivo encontrada (de 0 a 4000). No gráfico estão mostrados os valores das funções objetivos de todos os indivíduos da nuvem (para cada iteração). Pode-se perceber que a nuvem inicia com valores bem distintos e altos (até quase 4000) e caminha para um valor um pouco acima de 1500, que é um valor ótimo local ou global.

Figura 3 – MOVIMENTAÇÃO DA FUNÇÃO OBJETIVO DOS INDIVÍDUOS DA NUVEM



2.2.3 PSO Discreto e Binário

O PSO, na forma como foi apresentado por KENNEDY; EBERHART (1995), possui valores contínuos para sua atualização. A velocidade é atualizada com valores reais e aleatórios, o que faz com que seja retornado um valor real. Dessa maneira, o PSO é contínuo, mas ainda pode ser estudado um PSO discreto e, mais ainda, binário. Isso ocorre quando o vetor x possui valores inteiros. Assim, é necessária uma adaptação da atualização da velocidade para que essa

entrada seja binária. A seguir serão apresentados alguns artigos que abordaram o PSO discreto ou binário aplicado a um problema de produção.

KENNEDY; EBERHART (1997) propõem um PSO binário utilizando uma transformação logística, a função Sigmoide, de forma a limitar a transformação e estabelecer uma probabilidade (que será dada pelo vetor velocidade v) de mudança de valor do vetor posição x. Os autores utilizam a função combinada com uma distribuição normal para determinar se cada entrada do vetor x manterá seu valor ou não. KENNEDY; EBERHART (2001) unem os trabalhos dos autores de maneira mais explicativa. Os capítulos iniciais abordam assuntos como a Inteligência Artificial, organismo social, traz algumas outras metaheurísticas e também sobre o comportamento humano. Na segunda parte do livro é feita uma abordagem descritiva do tema principal (PSO), apresentando também variações, aplicações e implicações desta metaheurística. Tanto o artigo como o livro propõem a utilização da sigmoide S como função de avaliação da velocidade da partícula. Tal função é aplicada em metaheurísticas e redes neurais (por esses e outros autores) um pouco modificada, da seguinte forma:

$$S(v) = \frac{1}{1 + e^{-v}}$$

Dado um número randômico ρ obtido de uma distribuição normal entre 0 e 1, a avaliação do resultado e decisão da mudança ou não da variável de posição x é dada por:

$$x = \begin{cases} 1, \text{ se } \rho < S(v) \\ 0, \text{ caso contrário} \end{cases}$$

PAN; TASGETIREN; LIANG (2008) usam um algoritmo de PSO discreto, chamado de *Discrete Particle Swarm Optimization* (DPSO), para resolver um problema de sequenciamento de produção *Flow Shop* sem espera, ou seja, problema no qual não é permitido que a máquina fique ociosa entre a produção de duas tarefas de um produto em um mesmo período.

TASGETIREN; LIANG (2003) apresentam uma aplicação de um PSO binário em um problema de dimensionamento de lotes. Para transformar em um vetor binário, o vetor x, que representa a partícula, recebe o valor 1 se a decisão de dimensionamento de lotes é dada e 0 caso contrário. Ou seja, se há produção, $x = 1$. O vetor Q recebe a informação do tamanho do lote da partícula. A atualização da velocidade v é também restrita por uma função sigmoide, como a realizada pelos outros autores previamente citados. Os autores ainda colocam que, dentro de seu conhecimento, é a primeira aplicação reportada de um PSO binário aplicado em um problema de dimensionamento de lotes.

KASHAN; KARIMI (2009) utilizam uma distribuição Bernoulli para definir os parâmetros dos termos social e cognitivo. Com o uso do DPSO e um algoritmo de busca local, a representação dos vetores posição e velocidade foi estendida de um vetor real para um vetor inteiro, de forma que foi possível a aplicação em um problema de sequenciamento com máquinas paralelas.

LIAO; TSENG; LUARN (2007) desenvolvem um PSO discreto também utilizando a função sigmoide, proposta por KENNEDY; EBERHART (1997), para um problema de sequenciamento *Flow Shop*. Utilizam a mesma avaliação que TASGETIREN; LIANG (2003) para o vetor x. Os autores colocam também que a quantidade de aplicações de um algoritmo PSO discreto é muito pequena em comparação com as aplicações de PSO contínuo.

KULKARNI et al. (2015) fazem uma revisão de várias variações do PSO: híbrido PSO, adaptado, multiobjetivo e discreto. Citam também algumas aplicações do PSO em diversos problemas relacionados a engenharia mecânica.

Apesar de apresentar vários artigos relacionados ao PSO discreto e binário, não foram encontrados artigos que tratassem de um problema de planejamento de produção com preservação da preparação. É notável o aumento da dificuldade do problema para a aplicação da metaheurística adaptada a valores discretos e, por isso, tem-se uma motivação para o desenvolvimento deste trabalho.

2.3 METAHEURÍSTICAS APLICADAS EM PROBLEMAS DE PRODUÇÃO

São várias as metaheurísticas aplicadas em problemas de planejamento de produção e dimensionamento de lotes. TOLEDO et al. (2009) resolvem o problema de dimensionamento de lotes e programação da produção com e sem máquinas paralelas, e com penalização para demandas não atendidas, por três diferentes Metaheurísticas: Busca Tabu, *Simulated Annealing* e Algoritmo Genético. Os autores ainda comparam seus resultados computacionais com o *Threshold Accepting* (TA) proposto por FLEISCHMANN; MEYR (1997). No artigo TOLEDO; FRANÇA; MORABITO (2006), os autores resolvem um problema de dimensionamento de lotes e programação da produção com máquinas paralelas de uma indústria de bebidas, com restrição de capacidade, custos e tempos de *setup* dependentes da sequência de produção. Ainda é feita a aplicação do Algoritmo Genético Multi-Populacional, considerado um algoritmo genético modificado. Encontram-se muitos artigos que aplicam uma metaheurística em problemas de dimensionamento de lotes e programação da produção. Existem também muitos artigos de aplicação de metaheurísticas em problemas de *Flow Shop* e *Job Shop*.

Da aplicação do PSO em problemas de dimensionamento de lotes e programação da produção tem-se o já citado TASGETIREN; LIANG (2003), que utiliza um PSO binário. Além desses autores, CHAKRABORTTY et al. (2015) aplicam uma adaptação do PSO (um ambiente probabilístico) em um *Aggregate Production Planning* (APP), que é um problema de produção agregado com previsão de demanda. O problema trabalha com demanda futura, para um horizonte de planejamento de até 18 meses. WANG; YEH (2014) aplicam um PSO modificado também em um APP. Os autores citam que é feita uma adaptação que introduz o conceito de sub-partículas e um procedimento operacional modificado de atualização para regular os processos de busca da metaheurística. Esses dois últimos problemas são mais próximos de um problema real e foram aplicados em indústrias.

As aplicações em outros problemas de produção são várias. LIU; WANG; JIN (2008) fazem a aplicação de um PSO híbrido para um problema de *Flow Shop*, enquanto JARBOUI et al. (2008) fazem a aplicação de um PSO combinatório para problemas de *Flow Shop* permutacional. BANK et al. (2012) aplicam os algoritmos de PSO e *Simulated annealing* (SA) em problemas de *Flow Shop* nos quais o tempo de processamento depende do tempo de espera de antes do processo iniciar. SHA; LIN (2010) aplicam um PSO multi-objetivo, discreto, em um problema *Job Shop* e ZHAO et al. (2014) utilizam um PSO com um índice de perturbação de declínio, que serve para aprimorar a busca e reduzir a probabilidade de ficar preso em um ótimo local, em um problema de *Job Shop* multi-objetivo.

2.4 *MATH*-HEURÍSTICA

Com a dificuldade em resolver problemas de produção unicamente com metaheurísticas — muitas vezes por tempo computacional ou dificuldade com a linguagem de programação — tem-se usado uma alternativa que vem sendo eficaz: a combinação de um método matemático exato com uma metaheurística. (OBAL, 2016)

Essas aplicações vêm sendo chamadas de *Math*-heurísticas, pela combinação de heurísticas com um modelo exato.

As *Math*-heurísticas são heurísticas híbridas que utilizam a metaheurística e técnicas de programação matemática. A união entre a programação matemática e o campo da metaheurística está cada vez melhor, com a existência de excelentes *solvers* e com o objetivo de resolver os problemas da melhor forma possível, não mais com o objetivo de promover a metaheurística em questão.

OBAL (2016) ainda coloca que a combinação desses dois meios de resolução de problemas pode ser vista de duas maneiras: como uma forma de utilizar a programação matemática para melhorar resultados das metaheurísticas, ou utilizar metaheurísticas para melhorar resultados da programação matemática. Neste trabalho foi utilizada a primeira delas, utilizando a programação matemática para melhorar os resultados obtidos.

3 PROPOSTA DE APLICAÇÃO DE UMA *MATH*-HEURÍSTICA PARA RESOLVER UM PROBLEMA DE PRODUÇÃO

Neste capítulo será apresentada a proposta de *Math*-heurística aplicada ao problema de produção escolhido. Será descrito o modo como foi aplicado o PSO e de que maneira a metodologia exata é usada na aplicação da metaheurística.

3.1 INTRODUÇÃO

Com a finalidade de aplicar o problema na produção de uma indústria de tijolos, foram estudados alguns modelos de dimensionamento de lotes e programação da produção para uma exemplificação. Os problemas estudados são de pequeno porte, de forma que suas soluções ótimas são encontradas pela própria metodologia exata, para uma comparação.

Para o problema estudado foi utilizada uma *math*-heurística. Nessa proposta foi aplicado o PSO, programado no Matlab 7.9.0, juntamente com a metodologia exata *Branch-and-Bound*, codificado no Lingo 13.0, de forma que o otimizador faz a distribuição das quantidades após o PSO movimentar o vetor de decisão de produção.

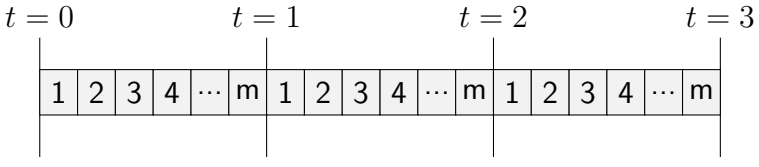
Para encontrar a população inicial do nosso problema será utilizada uma heurística proposta por FLEISCHMANN; MEYR (1997), chamada *Greedy-Mod*, que será explanada na seção a seguir.

3.2 *GREEDY-MOD*

A ideia é estimar a capacidade que pode ser consumida por um produto j nos períodos anteriores ao macro-período t quando o estado de preparação é fixo (FLEISCHMANN; MEYR, 1997). O *Greedy-Mod* é uma heurística muito utilizada para se obter as quantidades a serem produzidas com uma demanda pré-determinada.

Supondo que o problema contenha um horizonte de planejamento com T períodos divididos em m micro-períodos, conforme a Figura 4.

Figura 4 – MICRO-PERÍODOS



O algoritmo funciona da seguinte forma:

- Agregar os micro-períodos em cada período

- Iniciar:
  - Produção do período igual a zero
  - Verificar a quantidade de micro-períodos que têm produção em cada período
  - Adicionar a demanda de todos os períodos, para cada item.
  - Para cada período, do último ao primeiro, fazer:
    - Para todo item que tem produção do período maior que zero:
      - Adicionar o lote mínimo de produção
      - Atualizar capacidade
      - Atualizar demanda (retirar o mínimo entre: quantidade de produção; e demanda do período atual + demanda dos períodos seguintes)
      - Se a capacidade ficar negativa, parar o programa (infactível).
    - Senão, quantidade a produzir é nula.
    - Para todo item com produção do período maior que zero:
      - Se a demanda é maior do que a quantidade que a capacidade permite produzir, então
      - A diferença será o mínimo entre: a diferença entre a demanda e a quantidade possível de ser produzida $\left(\frac{\text{capacidade}}{\text{tempo de produção}}\right)$; a quantidade possível de ser produzida; e a demanda do período atual + demanda dos períodos seguintes.
      - Atualiza a quantidade a ser produzida: quantidade anterior + diferença
      - Atualiza capacidade
      - Atualiza demanda
    - Senão, quantidade a produzir é nula.
    - Para todo item com produção do período maior que zero, em ordem decrescente de $\frac{\text{custo de estoque}}{\text{tempo de produção}}$:
      - A diferença será o mínimo entre: a demanda do período atual + demanda dos períodos seguintes; e $\frac{\text{capacidade}}{\text{tempo de produção}}$.
      - Atualiza a quantidade a ser produzida: quantidade anterior + diferença
      - Atualiza capacidade
      - Atualiza demanda
    - Senão, quantidade a produzir é nula.
- Desagregar os períodos em micro-períodos, separando a quantidade de itens a serem produzidos igualmente entre os micro-períodos que possuem produção.

O *Greedy-Mod* proposto por FLEISCHMANN; MEYR (1997) faz a distribuição de quantidade de itens a serem produzidos para um problema de planejamento sem preservação da preparação. Considera que a matriz decisão de produção y de todos os itens já está definida. Portanto, para a aplicação no problema proposto neste trabalho é necessária uma modificação do método, levando em conta a produção em qualquer período (ou seja, todos os itens podem ter *setup*). Esta proposta é apresentada na Seção 3.3.

3.3 PROPOSTA DE APLICAÇÃO

Uma solução geral para o modelo dado pelas equações de 2.11 - 2.19, como já descrita anteriormente, será da forma

$$Q = \begin{bmatrix} q_{11} & \cdots & q_{1T} \\ \vdots & \ddots & \vdots \\ q_{N1} & \cdots & q_{NT} \end{bmatrix} = (q_{it})_{N \times T},$$

onde Q é a matriz das quantidades produzidas de cada item i em cada período t.

A partir da matriz de quantidades é gerada uma matriz de decisão, denominada aqui por DP, que será a matriz a ser atualizada pelo PSO. Essa matriz terá três tipos de entradas: receberá -1 para o item i no período t caso a máquina não tenha sido preparada nem vá produzir esse item nesse período; receberá 0 para o item i no período t caso a máquina tenha sido preparada para esse item no período $t-1$ e sua preparação foi mantida para o período t, iniciando sua produção pelo item i; e receberá 1 para o item i no período t caso a máquina tenha custo de *setup* para o item i e produza esse item no período t. A matriz DP terá a forma

$$DP = \begin{bmatrix} dp_{11} & \cdots & dp_{1T} \\ \vdots & \ddots & \vdots \\ dp_{N1} & \cdots & dp_{NT} \end{bmatrix} = (dp_{it})_{N \times T},$$

onde a variável dp_{it} é definida da seguinte forma:

$$dp_{it} = \begin{cases} -1, & \text{se não há produção do item } i \text{ no período } t \\ 0, & \text{se há produção do item } i \text{ no período } t \text{ mas não há custo de } \textit{setup} \\ 1, & \text{se há produção e custo de } \textit{setup} \text{ do item } i \text{ no período } t \end{cases}$$

A opção da não utilização da matriz Q com o PSO é devida a uma dificuldade encontrada na aplicação da metaheurística nessa matriz. O problema abordado de planejamento de produção é um problema no qual as quantidades são modificadas livremente. Se um item é produzido em um determinado período, existe uma quantidade que deve ser produzida deste item. Dessa forma, a matriz a ser movimentada pelo PSO não deve ser a matriz de quantidades e sim a matriz de decisão. Quando trabalha-se com a matriz de quantidades, se percebe uma enorme quantidade de infactibilidades devido à natureza do problema. Ao trabalhar-se com a

matriz de decisão tem-se uma quantidade menor de infactibilidades, mas ainda assim grande. Isso ocorre devido a maneira como o PSO é movimentado.

Um exemplo de matriz de decisão, supondo 3 itens e 4 períodos de produção, é:

$$DP = \begin{bmatrix} 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Nesse exemplo, o item 1 é produzido e tem *setup* nos períodos 1 e 2, não tem *setup* no período 3 e não tem produção no período 4. O item 2 tem *setup* nos períodos 1 e 3 e não tem nos períodos 2 e 4. O item 3 tem *setup* nos períodos 1 e 4 e não tem produção nos períodos 2 e 3.

Para aplicar a metaheurística no problema deve-se, primeiramente, criar uma população inicial. A seção a seguir traz algumas maneiras de criar uma população inicial utilizando algumas heurísticas.

3.3.1 População Inicial

Primeiramente, é criada uma população inicial que será movimentada de acordo com o PSO, modificando a população. Essa população deve conter indivíduos diferentes, para que a diversidade das soluções seja maior, ou seja, não caia em mínimos ou máximos locais. Nesse problema, em particular, serão criados seis tipos diferentes de indivíduos, gerados de maneiras diferentes, detalhados a seguir.

3.3.1.1 1º Tipo de Indivíduo: Relação Estoque/Produção (*Backward*)

A matriz de quantidades $Q = (q_{it})_{N \times T}$ é preenchida de acordo com a heurística abaixo:

- Para cada período, variando de T a 1:
  - Para escolha em ordem decrescente de $\frac{\text{Custo de Estoque } (h_i)}{\text{Tempo de Produção } (b_i)}$ dos itens i a serem produzidos:
    - Verifica se o item possui demanda naquele período;
    - Se sim, o número de itens a serem produzidos será

$$diff = \min\left\{Dem_{it}, \frac{C_t}{b_i}\right\},$$

onde $Dem_{it} = \sum_{\tau=1}^{T} d_{i\tau} - \sum_{\tau=t}^{T} q_{i\tau}$ é a demanda ainda não atendida do item i (no período t e nos períodos seguintes) , $\frac{C_t}{b_i}$ é a quantidade de itens i que podem ser produzidos no período t.

* Atualiza a matriz de quantidades Q, a capacidade disponível C_t e a demanda restante a ser atendida Dem_{it}.

3.3.1.2 2º Tipo de Indivíduo: Relação Estoque/Produção (*Forward*)

A matriz de quantidades é preenchida de acordo com a heurística abaixo:

- Para cada período, variando de 1 a T:
  - Preenche a demanda não atendida do item neste período;
  - Itens i em ordem crescente de $\frac{\text{Custo de Estoque } (h_i)}{\text{Tempo de Produção } (b_i)}$:
    * Para os itens que ainda não estão sendo produzidos deve-se antes retirar o Tempo de Preparação do item da Capacidade do período.
    * O número de itens a mais será

$$diff = \min\left\{Dem_{it}, \frac{C_t}{b_i}\right\},$$

onde $Dem_{it} = \sum_{\tau=1}^{T} d_{i\tau} - \sum_{\tau=1}^{t-1} q_{i\tau}$ é a demanda ainda não atendida do item i, $\frac{C_t}{b_i}$ é a quantidade de itens i que podem ser produzidos no período t.

    * Atualiza a matriz de quantidades Q, a capacidade disponível C_t e a demanda restante a ser atendida Dem_{it}.

3.3.1.3 3º Tipo de Indivíduo: Maior custo de estoque (*Backward*)

A matriz é preenchida de acordo com os seguintes passos:

- Para cada período, variando de T a 1:
  - Itens i em ordem decrescente de Custo de Estoque:
    * Para os itens que ainda não estão sendo produzidos deve-se antes retirar o Tempo de Preparação do item da Capacidade do período.
    * Preenche demanda não atendida dos períodos posteriores.
    * Preenche o máximo possível da demanda do período.
    * Atualiza a matriz de quantidades Q e a capacidade disponível C_t.

3.3.1.4 4º Tipo de Indivíduo: Menor custo de estoque (*Forward*)

A matriz de quantidades é preenchida da seguinte forma:

- Para cada período, variando de 1 a T:
  - Itens i em ordem crescente de Custo de Estoque:
    - Para os itens que ainda não estão sendo produzidos deve-se antes retirar o Tempo de Preparação do item da Capacidade do período.
    - Preenche com a demanda do período.
    - Atualiza a matriz de quantidades Q e a capacidade disponível C_t.
    - Caso a capacidade ainda seja diferente de 0, acrescentar unidades a mais na produção (seguindo a ordem crescente de Custo de Estoque).

3.3.1.5 5º Tipo de Indivíduo: Demanda passada (*Backward*)

Para o 5º indivíduo da população, será considerada a demanda passada. A matriz de quantidades será encontrada da seguinte forma:

- Para cada período, variando de T a 1:
  - Se o período for diferente de 1, acrescenta Tempo de *Setup* na Capacidade do período.
  - Em ordem decrescente (diferente de zero) de

$$\frac{\text{Demanda não atendida período atual + posteriores}}{\text{Demanda restante acumulada}},$$

    para cada item:
    - Preenche com a quantidade máxima de itens até produzir toda a demanda do período e demanda de períodos posteriores (que ficarão em estoque pois a capacidade do período não foi suficiente).
    - Atualiza capacidade C_t e matriz de quantidades Q.

3.3.1.6 6º Tipo de Indivíduo: Demanda futura (*Forward*)

O 6º e último indivíduo da população terá sua matriz de quantidades encontrada com os passos abaixo:

- Para cada período, variando de 1 a T:

– Em ordem decrescente (diferente de zero) de

$$\frac{\text{Demanda não atendida período atual + posteriores}}{\text{Demanda restante acumulada}},$$

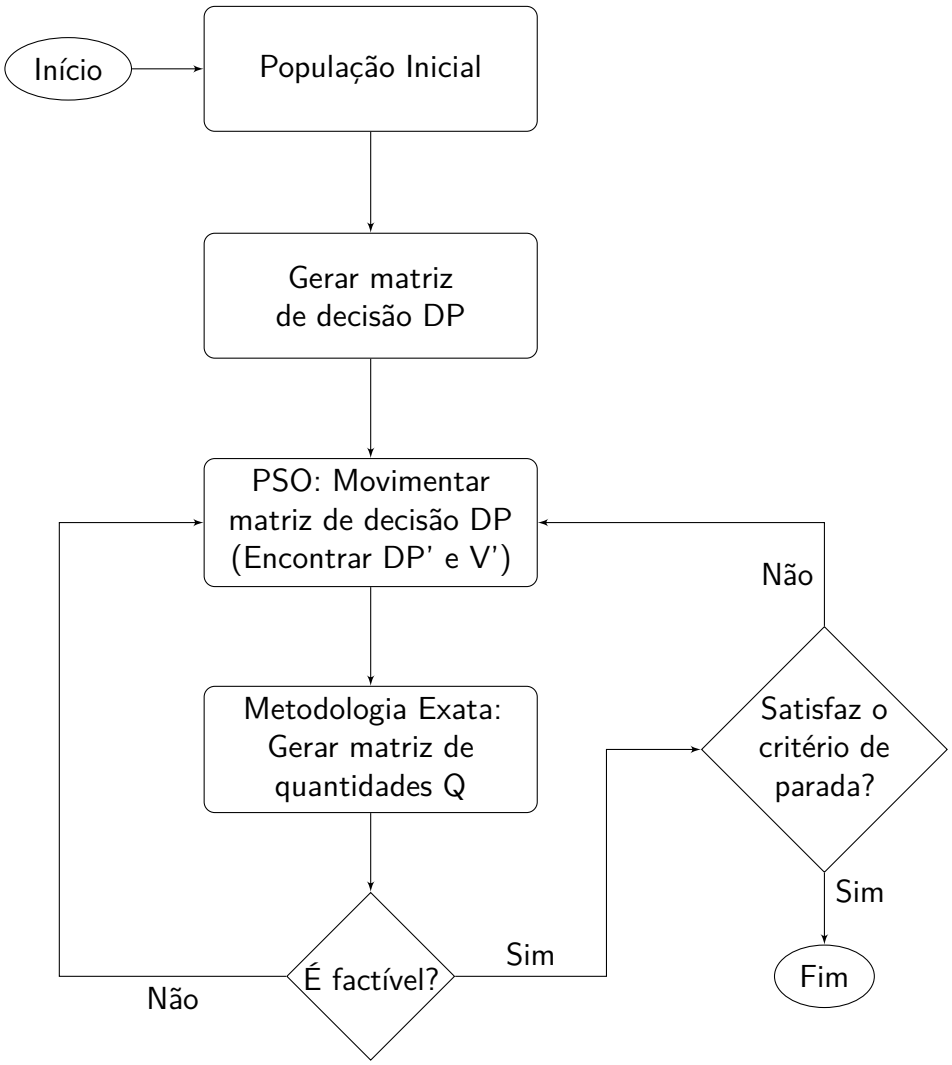
para cada item:

* Preenche com a quantidade máxima possível da demanda do período.
* Recomeça preenchendo com a quantidade adicional de itens a serem produzidos (para estoque).
* Atualiza capacidade C_t, matriz de quantidades Q, demanda restante acumulada e demanda não atendida.

3.3.2 Adaptação do PSO e Codificação

A Figura 5 traz um fluxograma que mostra como o algoritmo proposto funciona e, a seguir, é explicado detalhadamente cada parte do processo.

Figura 5 – FLUXOGRAMA



Após criar a população inicial deve-se movimentar a solução para encontrar uma nova posição dessa população. Essa movimentação é realizada com um PSO modificado, visto que a matriz ao qual será aplicado é a matriz de decisão DP.

A movimentação será realizada de acordo com o resultado da entrada da matriz de velocidades V, matriz que será movimentada utilizando a posição atual, velocidade atual, *pbest* e *gbest*, além de alguns parâmetros e números aleatórios determinando os pesos de cada um desses valores. A matriz V é adicionada à matriz de decisão DP. Caso essa adição resulte em um valor menor ou igual a um valor negativo α, será trocada para -1. Caso resulte em um valor maior que α e menor ou igual a um valor positivo β, será trocada para 0 (quando possível). Caso resulte em um valor maior que β, será trocada para 1. Os valores α e β devem ser ajustados de forma que as infactibilidades sejam reduzidas. Os testes computacionais consideram parâmetros previamente definidos e testados, devendo estes serem os melhores valores para se obter uma solução ótima ou mais perto da ótima. Atualiza-se a matriz V para a continuidade do processo.

$$v_{it} = \begin{cases} 1, & \text{se } dp_{it} > \beta \\ 0, & \text{se } \alpha < dp_{it} < \beta \text{ (quando possível)} \\ -1, & \text{se } dp_{it} < \alpha \end{cases}$$

Após essa movimentação, é realizada uma mutação para que as soluções não caiam em mínimos ou máximos locais. Aleatoriamente escolhem-se o item e o período a sofrerem mutação. Obtendo e analisando-se um número aleatório é verificado de que forma será alterado o valor da matriz DP para aquele item e período previamente definidos. Caso o número aleatório seja maior que um valor δ (pré-determinado), o valor da entrada da matriz DP será alterada para 1. Se for menor que um valor γ (pré-determinado e negativo), o valor da entrada será alterado para -1. No intervalo de γ a δ será alterado para 0 (quando o item teve produção no período anterior).

Depois de feita a atualização de DP, deve ser verificada a preservação da preparação de produção dessa matriz. Ou seja, a matriz de decisão pode ter sido movimentada de forma que dois itens tenham sua preparação mantida de um período a outro. Isso deve ser corrigido para evitar infactibilidades.

A codificação foi adaptada para que pudesse ser aplicado o PSO nesse tipo de problema. A matriz de decisão DP já foi previamente definida, com valores discretos. A matriz Q possui valores reais e as matrizes *pbest* e *gbest* possuem as melhores matrizes de decisão, ou seja, possuem entradas inteiras assim como a matriz DP, já que são as melhores soluções encontradas para DP (individualmente ou em grupo).

Para encontrar a matriz de quantidades Q foi utilizada uma metodologia exata com a matriz DP já definida. Com a matriz de quantidades Q já determinada, obtem-se por meio de um otimizador um valor de *fitness* que possibilita avaliar a qualidade da solução. Esta

solução pode ser um *pbest*, *gbest* ou também uma solução pior do que algumas já encontradas anteriormente. É importante que os parâmetros que definem os pesos do termo cognitivo (*pbest*) e do termo social (*gbest*) sejam testados para que a movimentação da população não a encaminhe para um máximo ou mínimo local. Tais parâmetros devem ser muito bem escolhidos para que a otimização seja a melhor possível.

Com as matrizes DP e Q pode-se verificar a factibilidade do problema. Caso o problema seja factível, é verificado se satisfaz o critério de parada. Caso a resposta seja positiva, é encerrada a movimentação. Em caso de resposta negativa à factibilidade ou ao critério de parada, deve-se movimentar novamente a matriz de decisão DP com a ajuda do PSO até que o critério de parada seja satisfeito. Este critério pode ser definido com base em diversos fatores: tempo de processamento, quantidade de iterações, quantidade de iterações nas quais o *fitness* se repete, entre outros.

3.4 TESTE DE FUNCIONALIDADE

A proposta para a aplicação do PSO híbrido no modelo de programação da produção é de que seja aplicada a metaheurística em um vetor que possui entradas de valores discretos com a atualização desses valores enviada para um otimizador distribuir as quantidades.

Para mostrar a funcionalidade da proposta, o primeiro problema a ser estudado possui 3 itens e 4 períodos, e os dados descritos nas tabelas 1, 2, 3 e 4. O exemplo utilizado é o descrito em ARENALES et al. (2007).

Tabela 1 – DEMANDA

PERÍODO / ITEM	PERÍODO 1	PERÍODO 2	PERÍODO 3	PERÍODO 4
ITEM 1	1	10	3	10
ITEM 2	2	4	0	5
ITEM 3	2	4	0	5

Tabela 2 – CUSTOS DE PRODUÇÃO E ESTOQUE (POR ITEM)

	CUSTO DE PREPARAÇÃO	CUSTO DE ESTOQUE
ITEM 1	350	150
ITEM 2	100	100
ITEM 3	90	70

Tabela 3 – TEMPOS DE PRODUÇÃO E PREPARAÇÃO (EM HORAS, POR ITEM)

	TEMPO DE PRODUÇÃO	TEMPO DE PREPARAÇÃO
ITEM 1	20	40
ITEM 2	10	40
ITEM 3	20	40

Tabela 4 – CAPACIDADE DO PROBLEMA (EM HORAS)

PERÍODO 1	PERÍODO 2	PERÍODO 3	PERÍODO 4
280	320	280	400

A função objetivo, encontrada pelo otimizador Lingo, quando considerado o problema de planejamento da produção com preservação da preparação, é de 1585 unidades monetárias. Assim, a tentativa é de otimizar este problema programando o PSO e um sistema computacional de otimização e chegar a este mesmo resultado, em se tratando de um problema de pequeno porte.

A matriz de quantidades do problema é a matriz que fornece o valor da função objetivo e, portanto, o objetivo da otimização é encontrar os melhores valores possíveis para essa matriz. A aplicação do PSO, porém, não será feita na matriz de quantidades e sim na matriz de decisão.

Primeiramente, determinam-se os seis indivíduos da população como descritos na Seção 3.2.1. O PSO é então aplicado nesses indivíduos de forma a movimentar a nuvem para encontrar outros valores de *fitness* e melhorar a solução.

A tabela 5 mostra os resultados obtidos com a programação do PSO em um sistema computacional com a ajuda do Lingo para distribuir as quantidades a serem produzidas. O valor da Função Objetivo é o menor valor encontrado para o problema, para cada metodologia.

Tabela 5 – TESTE DE FUNCIONALIDADE

ITENS X PERÍODOS	FUNÇÃO OBJETIVO	METODOLOGIA	TEMPO COMPUTACIONAL (EM SEGUNDOS)
3x4	1585	PSO + Exata	1900
3x4	1585	Exata	1

A matriz de decisão que gerou a função objetivo de valor 1585 encontrada para este problema foi

$$DP = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix},$$

com as quantidades distribuídas da forma

$$Q = \begin{bmatrix} 1 & 10 & 4,5 & 8,5 \\ 2 & 4 & 0 & 5 \\ 6 & 0 & 0 & 5 \end{bmatrix}.$$

Neste caso, de acordo com a matriz DP, houve preservação da preparação do primeiro ao segundo período para o item 2 e do segundo ao terceiro e do terceiro ao quarto para o item 1. As entradas -1 representam que não houve produção, o que pode ser verificado em

também Q. O valor 1585 da função objetivo foi encontrado com os parâmetros $w = 0,3$, $\eta_1 = 0,3$ e $\eta_2 = 0,3$ para os pesos dos termos de inércia, cognitivo (*pbest*) e social (*gbest*), respectivamente, $\alpha = -0,4$ e $\beta = 0,8$ como parâmetros na atualização da velocidade, e $\delta = 0,7$ e $\gamma = -0,4$ como parâmetros da mutação. Estes parâmetros foram escolhidos através de testes, de forma que a convergência não fosse muito lenta ocasionando um grande tempo computacional ou também muito rápida caindo em mínimos locais.

Pode-se verificar que o tempo computacional dos dois programas é muito distinto. Uma dificuldade encontrada para esse modelo é que a matriz de quantidades não pode ser a matriz a ser movimentada, devido a natureza de seus valores de entrada. Por isso foi necessária a movimentação da matriz de decisão DP mas, para a obtenção do valor *fitness*, é necessária a distribuição das quantidades, que foi realizada pelo Lingo. Para uma matriz DP podem ser determinadas inúmeras matrizes Q, mas a desejada é a matriz Q que contém o menor valor do *fitness*. Essa distribuição precisa ser realizada para cada indivíduo da população em cada iteração, fazendo com que seja necessária a chamada do Lingo todas as vezes, o que torna lento o processo de obtenção de uma solução.

O valor da função objetivo pode variar de acordo com os parâmetros inseridos pois são estes que fazem com que o valor *fitness* não permaneça em ótimos locais. Como a programação contém muitos valores aleatórios, a função ótima não foi encontrada em todas as vezes que o programa foi executado.

Neste problema de pequeno porte foi encontrada a solução ótima mas, ainda assim, não foi eficiente a aplicação do PSO proposta. Tal aplicação se mostrou inviável para esse problema quando analisado o tempo computacional, que foi muito elevado em comparação à resolução pela metodologia exata. Espera-se, entretanto, que esta metodologia apresente melhores resultados caso novos testes e escolha mais conveniente de novos parâmetros sejam feitos.

4 APLICAÇÃO DA PROPOSTA EM UMA INDÚSTRIA DE TIJOLOS

4.1 PROBLEMA

Uma indústria que possua uma linha de produção, como uma indústria de tijolos (olaria), muitas vezes precisa de um engenheiro de produção ou alguém que faça uma otimização de todos os processos pelo qual o produto passa antes de chegar ao produto final. Tal otimização pode ser realizada por meio da resolução de problemas com modelos matemáticos de pesquisa operacional, como é o caso do problema de dimensionamento de lotes trabalhado.

São várias as etapas de produção de uma olaria até a obtenção do tijolo. Após a argila sair do depósito, tem-se a preparação da argila, a linha de produção (retirada de pedaços de madeira e metais, retirada de ar da massa e produção dos tijolos com uma matriz), o corte das peças, secagem natural e a queima em forno. Todas as etapas descritas anteriormente são realizadas em sequência. A fabricação do tijolo é, portanto, uma linha de produção que não pode ter sua ordem modificada.

Um problema encontrado nessa indústria é o tempo de espera para a secagem natural do tijolo. Essa espera pode ser de 7 a 15 dias, dependendo das condições climáticas. Além disso, muitas vezes é necessária a busca de alternativas para aumentar ou diminuir a velocidade dessa secagem. Se a temperatura está alta, é preciso instalar ventiladores para refrescar o ambiente de forma que os tijolos não sequem muito rápido (pois isso causariam rachaduras). Assim, se a temperatura estiver baixa, o tijolo seca muito devagar e causaria um atraso na produção por alguns dias (por lotação do espaço disponível para estocagem).

Além do tempo de secagem, pode-se ter também a demanda como problemática para essa indústria. Essa variável pode ser muito dinâmica. Em algumas épocas estarão sendo produzidos os tijolos já encomendados (quando há pedidos em grandes quantidades). Mas em outras ocasiões pode-se estar produzindo para preencher o estoque. Dessa maneira, existe a necessidade da previsão do que é mais interessante ser mantido em estoque, ou seja, qual a quantidade de cada tijolo que, mantendo em estoque, não demorará muito a ser requisitado. Essa preocupação com o tempo de estoque se deve ao fato de que além dessas duas problemáticas, existe uma norma que estipula um tempo máximo que o tijolo pode ficar estocado.

Diante de todos esses problemas encontrados em uma produção de tijolos, decidiu-se formular e resolver um modelo matemático que otimiza a produção, secagem e estocagem dos tijolos. Como a demanda varia com o tempo, a produção visa atender à demanda ainda não atendida e também ao preenchimento do estoque, até o seu limite e seguindo o histórico de pedidos.

O modelo formulado é de dimensionamento de lotes e planejamento de produção, com

preservação da preparação e capacidade. A produção dos tijolos é feita com uma matriz (que varia conforme o tipo do tijolo) e também há mais de uma opção para o corte. Dessa maneira, as máquinas podem estar preparadas para um determinado tipo de tijolo, e a mudança para outro tipo leva um certo tempo (que está sendo considerado como o tempo de preparação). Existe a preservação dessa preparação pois a máquina pode ter trabalhado em um formato de tijolo até o final de um dia e iniciar o mesmo tijolo no dia seguinte, sem precisar preparar a máquina novamente. A capacidade depende da quantidade de argila disponível, quantidade de horas dos funcionários e tamanho do galpão para secagem e estoque.

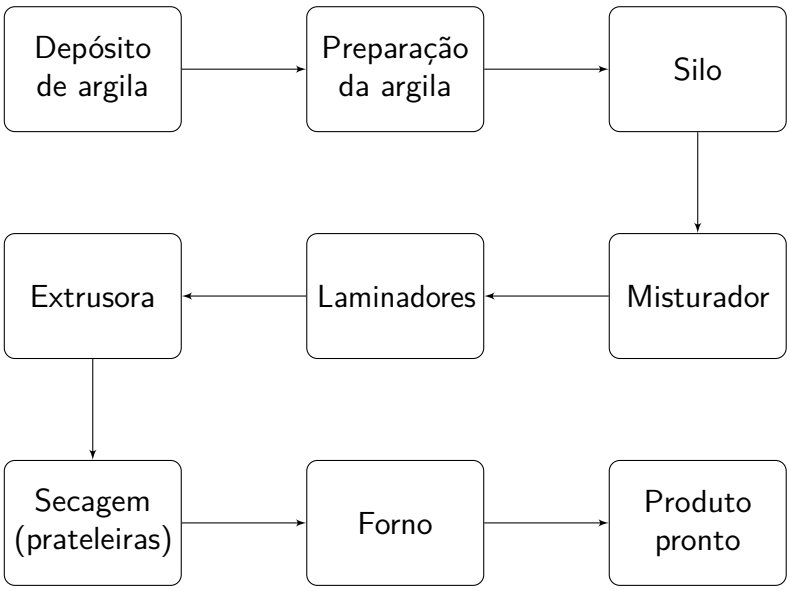
4.2 FUNCIONAMENTO DA INDÚSTRIA

Uma indústria de tijolos possui várias etapas desde a preparação da argila até o produto final. É importante conhecer cada parte da linha de produção para a escrita do modelo matemático a ser estudado. Uma olaria possui muitas vezes diversos tipos de itens e tamanhos, por isso o modelo matemático a ser estudado é um modelo com múltiplos itens. Além disso, possui uma capacidade (em tempo de horas de trabalho e também em quantidade de matéria-prima), demanda (variável dependendo do item e do período), tempo e custo de *setup* (troca da preparação da máquina do item i para o item j), tempo de produção de cada item e também custo de estoque.

Na indústria estudada existem 7 diferentes processos pelo qual a argila passa antes de ser comercializada. Se a argila não passar por todos esses processos o tijolo fabricado pode não ter uma boa qualidade, conter rachaduras, se desmanchar antes da secagem, ou até conter pedaços de galhos, rochas ou metais. Tudo isso faz com que o produto final seja desvalorizado, por isso a necessidade de várias máquinas para garantir uma melhor qualidade da argila na produção do tijolo. A seguir estão discriminados esses processos:

- Preparação da argila;
- Silo: alimentador;
- Misturador: mistura a argila com a terra;
- Laminadores: cilindros que quebram as pedras e uniformiza a argila;
- Extrusora: retira o ar da argila e dá forma ao tijolo;
- Secagem: prateleiras para a secagem natural do tijolo, que leva de 7 a 21 dias;
- Forno: queima do tijolo para a obtenção do produto final.

Figura 6 – SEQUÊNCIA DE PRODUÇÃO



4.2.1 Preparação da argila

A argila utilizada na fabricação do tijolo não vem livre de resíduos (pedras, galhos, pedaços de metais, pregos, etc.). Ela deve ser preparada antes da fabricação de forma que as máquinas não sejam prejudicadas, o tijolo produzido não seja perdido e a produção de tijolos comercializáveis seja a maior possível.

De um depósito, a argila é retirada e misturada com terra. Em seguida, é colocada no Silo, que é um caixão alimentador de material, que determina a quantidade de matéria-prima para a produção. As Figuras 7, 8 e 9 mostram as etapas de preparação da argila para a produção.

Figura 7 – DEPÓSITO DE ARGILA

4.2.2 Silo

O Silo é um maquinário que determina a quantidade de material que será utilizado para a produção. Nele é colocada a argila e a terra, ainda com resíduos e já misturados, para o

Figura 8 – ARGILA A SER MISTURADA E PREPARADA

Figura 9 – PREPARAÇÃO INICIAL DA ARGILA

início da produção. A Figura 10 mostra o Silo da indústria, que comporta 1 (uma) tonelada. É essa máquina que determina a quantidade que pode ser produzida diariamente, dependendo da quantidade de argila com terra colocada em seu interior.

Figura 10 – SILO

4.2.3 Misturador e Homogeneizador

Na sequência do Silo pode-se ver a máquina chamada de Caixão Misturador, que serve para misturar a argila com a terra de maneira uniforme. Entre as duas máquinas existe um imã

para impedir que pedaços de ferro e metal continuem na produção (Figura 11).

Figura 11 – IMÃ COM PREGOS RETIRADOS DA ARGILA

Ao chegar no misturador, a terra e a argila são misturadas de maneira uniforme. A Figura 12 mostra de que forma essa mistura é realizada. O misturador pode ser substituído por uma máquina com maior utilidade, um Homogeneizador. Essa máquina, por sua vez, elimina resíduos da argila. Funciona como uma peneira, impedindo que galhos e pedaços muito grandes de madeira, pedras e metal passem aos outros processos da produção. A limpeza desse homogeneizador deve ser feita uma vez por semana, em média, e demora cerca de 30 minutos.

Figura 12 – MISTURADOR

Após a argila passar pelo misturador existe um outro imã que, caso ainda contenha algum metal após essas primeiras etapas da produção, esse imã detecta esse metal e a produção é parada até que o metal seja removido, pois a partir desse ponto qualquer peça de ferro pode estragar uma das máquinas seguintes do processo (Figura 13).

4.2.4 Laminadores

Na sequência do Misturador ou Homogeneizador encontram-se os Laminadores. Os Laminadores servem para quebra de pedras e uniformização da umidade. São cilindros distantes 5mm ou 2mm entre si, que não permitem que pedaços muito grandes de resíduos e argila

Figura 13 – IMÃ QUE DETECTA PEDAÇOS DE METAL

passem para a máquina seguinte, a Extrusora. A argila passa primeiramente no Laminador de 5mm, para a quebra das pedras maiores (Figura 14). Em sequência passa pelo Laminador de 2mm, para a quebra das pedras menores que restaram.

Figura 14 – LAMINADOR DE 5mm

Após um tempo a máquina perde sua precisão e o espaçamento de 2mm ou 5mm. Isso ocorre devido à grande quantidade de argila que passa entre dois cilindros e à menor quantidade de argila que passa nas extremidades, deformando o cilindro e permitindo a passagem de pedras maiores. Quando isso acontece, é necessária a utilização da Retífica, que é uma máquina que funciona como uma lixa, para deixar o cilindro com a superfície reta novamente, mantendo a distância de 5mm ou 2mm.

4.2.5 Extrusora

Depois da passagem pelos Laminadores, a argila está quase pronta para ser moldada. A Extrusora é acoplaca à uma bomba de vácuo (Figura 16), que retira todo o ar da argila para que ela tenha liga e possa ser moldada sem se desfazer. Na sequência a matéria-prima passa pela Extrusora (Figuras 17 e 18), que no final possui a placa de molde e tem-se o produto final (sem corte).

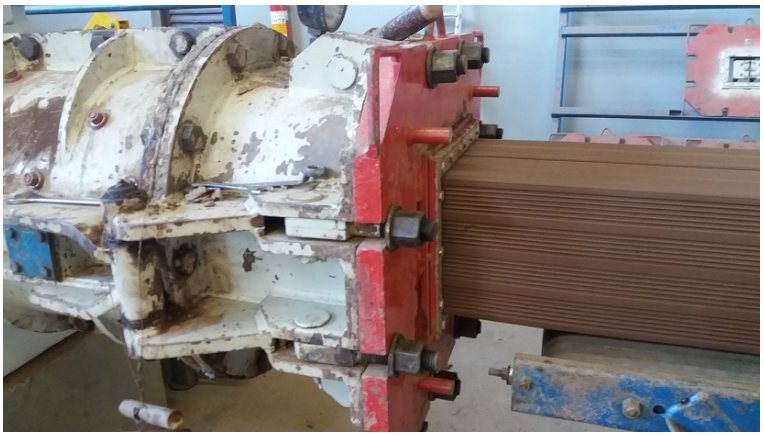
Figura 15 – LAMINADORES: 5mm (À ESQUERDA) E 2mm (À DIREITA)

Figura 16 – BOMBA DE VÁCUO ACOPLADA À EXTRUSORA

Figura 17 – EXTRUSORA

A Extrusora molda o tijolo de acordo com a placa que está fixada na saída da massa. A placa é alterada de acordo com a demanda e estoque de tijolos da fábrica. Existem 10 diferentes placas para a fabricação de tijolos: 6 furos, 8 furos, 9 furos, canaleta, entre outras (Figura 19). A mudança dessa placa é feita em aproximadamente 5 minutos e quando for necessária. Como a produção não deve acontecer durante a troca de placas, essa troca ocorre de forma a otimizar o processo. O corte também pode ser realizado de diferentes maneiras. O tijolo pode ser cortado com 19cm, 24cm ou 29cm. Para cada um desses tamanhos a máquina

Figura 18 – EXTRUSORA COM BLOCO DE TIJOLO JÁ MOLDADO (SEM CORTE)



de corte é montada de forma diferente. A adaptação da máquina de corte para o corte desejado é feita também em aproximadamente 5 minutos. A Figura 20 mostra o tijolo já moldado sendo cortado e a Figura 21 mostra o final da linha de produção, com os tijolos prontos para a secagem.

Figura 19 – PLACA DE MOLDE DO TIJOLO

Figura 20 – TIJOLO SENDO CORTADO

A fabricação na indústria em questão é de 18 a 21 toneladas por hora (quantidade depende do tipo do tijolo), sendo um total de 5 horas por dia trabalhado na produção dos

Figura 21 – TIJOLO MOLDADO E CORTADO PRONTO PARA A SECAGEM

tijolos.

4.2.6 Secagem

Assim que os tijolos são cortados, os funcionários da fábrica os colocam em prateleiras para a secagem natural do tijolo. O tijolo seca naturalmente de 7 a 21 dias, dependendo do clima. No outono e inverno esse tempo é normalmente de 21 dias e ainda é necessária a ajuda de ventiladores e aquecedores para acelerar o processo. O período mínimo de secagem (7 dias) é importante para que o produto não sofra rachaduras após a queima em forno.

Quando o clima está frio são utilizados ventiladores para dispersar o ar quente que sai do forno e acelerar o processo de secagem dos tijolos já prontos. Como o local onde esses tijolos são armazenados é limitado, se o tijolo demorar muito para secar não é possível produzir mais tijolos, atrasando entregas. Dessa forma, os ventiladores são muito utilizados para diminuir prejuízos por atraso de entregas, mas, por outro lado, o gasto de energia também aumenta em 30% (sendo que o custo da energia elétrica é maior das 18 às 21 horas e, por isso, a fábrica não utiliza os ventiladores nesses horários). As Figuras 22 e 23 mostram as prateleiras de tijolos secando naturalmente com a luz do sol e, em determinadas épocas, com a ajuda de ventiladores.

4.2.7 Forno

Os tijolos são verificados para saber se estão prontos para serem queimados. Essa verificação é feita de forma visual porque, dependendo do clima, o tempo para a secagem natural do tijolo varia muito. Caso um tijolo não seco seja colocado no forno, ele pode quebrar mais facilmente depois que terminar a queima.

O forno dessa indústria possui capacidade para aproximadamente 40 mil tijolos do tipo padrão (9x14x19cm), caso estejam no ponto para serem queimados. A queima inicialmente é para a retirada total da água da argila (que varia de 10 a 16 horas aproximadamente). Quando o forno chegar à $450\,^{\circ}\mathrm{C}$, toda a umidade já foi retirada e a temperatura é elevada até $810\,^{\circ}\mathrm{C}$

Figura 22 – TIJOLOS SECANDO NATURALMENTE ANTES DA QUEIMA, PARTE INTERIOR DO GALPÃO

Figura 23 – PARTE EXTERIOR DO GALPÃO ONDE OS TIJOLOS FICAM ARMAZENADOS ANTES DA QUEIMA

para terminar a queima (aproximadamente 4 horas). Depois dessas 4 horas ainda são esperadas de 6 a 7 horas para a retirada dos tijolos (resfriamento), para então ser realizada a troca por tijolos ainda não queimados.

Figura 24 – FORNO E TIJOLOS PRONTOS PARA SEREM QUEIMADOS

A Figura 24 mostra os tijolos já empilhados para serem inseridos no forno e queimados. Enquanto um vagão é colocado no forno, os tijolos já queimados do outro vagão são retirados

Figura 25 – ALIMENTAÇÃO DO FORNO COM LENHA

e esse vagão é preenchido com outros tijolos a serem queimados. O forno permanece aceso de 21 a 32 horas por cada queima. O combustível utilizado é lenha, inserida durante todo o tempo da queima. A Figura 25 mostra a parte do forno onde a lenha é inserida.

Figura 26 – PALETES DE TIJOLOS ESTOCADOS PARA A VENDA

Depois de retirados do forno e resfriados, os tijolos são colocados em paletes e estocados ou vendidos (a quantidade de tijolo em cada palete varia conforme o tamanho do tijolo). A venda na maior parte das vezes é realizada para construtoras, em grande escala.

4.3 MODELO

O problema de produção que será trabalhado para essa indústria é de dimensionamento de lotes e programação da produção. O modelo a seguir tem como base o modelo das equações (2.11) a (2.19), ou seja, contará com restrição de capacidade e preservação da preparação, mas com alguns dados e restrições a mais.

A primeira variável adicionada, $E0_{it}$, representa o estoque intermediário do item i no período t para um item que foi produzido há um período (faltando um período para a queima). $E0_{i0}$ (estoque intermediário do item i com um período) é condição inicial de $E0_{it}$ e representa a quantidade de itens em estoque intermediário (itens produzidos e ainda não queimados) que

estão com um período de secagem, faltando apenas um período para a queima. Tais itens serão entregues para cumprir a demanda do primeiro período, visto que serão queimados no fim deste primeiro período e já entregues.

A variável $E00_{it}$ representa o estoque intermediário do item i no período t para um item que foi recém produzido, ou seja, faltam dois períodos para a queima deste item. A condição inicial relacionada à essa variável é $E00_{i0}$ (estoque intermediário do item i recém produzido), que representa a quantidade de itens estocados em um estoque intermediário em que faltam dois períodos para a queima, ou seja, acabaram de ser produzidos. Ao final do primeiro período, estes itens passam a ser parte do estoque intermediário do item i com um período.

Abaixo serão definidos os índices, parâmetros, dados, condições iniciais e variáveis do problema e, na sequência, será apresentado o modelo aplicado. Existem algumas variáveis e condições iniciais diferentes do modelo dado pelas equações (2.11) a (2.19), devido à queima da cerâmica.

Índices:

i : itens

t : períodos

Parâmetros:

M : número grande

N : número de itens do problema

T : número de períodos

Dados:

b_i : capacidade (em unidades de tempo - u.t.) necessária do recurso da máquina para produzir uma unidade do item i

C_t : capacidade total (em u.t.) disponível do recurso da máquina no período t

d_{it} : demanda do item i atendido ao final do período t

h_i : custo de estoque do item i por período

s_i : custo de *setup* do item i

sp_i : tempo de *setup* (em u.t.) da máquina para processar o item i

Condições Iniciais:

$E0_{i0}$: estoque intermediário do item i com um período (com um período de secagem, ou seja, faltando um período para ir ao forno)

$E00_{i0}$: estoque intermediário do item i recém produzido (faltando dois períodos para ir ao forno)

I_{i0} : estoque inicial do item i

Variáveis:

I_{it} : estoque do item i no final do período t

$E0_{it}$: estoque temporário entre produção e queima da cerâmica para itens produzidos há um período (com um período faltando para a queima)

$E00_{it}$: estoque temporário entre produção e queima da cerâmica para itens recém produzidos (com dois períodos faltando para a queima)

q_{it} : quantidade do item i a ser produzida no período t

$$w_{it} : \begin{cases} 1, & \text{se o estado de preparação para o item } i \text{ continua do período } t-1 \text{ ao período } t \\ 0, & \text{caso contrário} \end{cases}$$

$$y_{it} : \begin{cases} 1, & \text{se ocorreu } \textit{setup} \text{ na máquina para o item } i \text{ no período } t \\ 0, & \text{caso contrário} \end{cases}$$

$$z_t : \begin{cases} 1, & \text{se não existe preparação de itens em } t, \text{ sendo que a preparação de um item específico ocorreu em } t-1 \text{ e é mantida até o período } t+1 \\ 0, & \text{caso contrário} \end{cases}$$

Formulação do Modelo:

$$min \sum_{i=1}^{N} \sum_{t=1}^{T} (s_i y_{it} + h_i I_{it}) \tag{4.1}$$

$$I_{i1} = I_0 + E0_{i0} - d_{i1} \qquad i = 1, \ldots, N \tag{4.2}$$

$$I_{it} = I_{i,t-1} + E0_{i,t-1} - d_{it} \qquad i = 1, \ldots, N \;\; e \;\; t = 2, \ldots, T \tag{4.3}$$

$$E0_{i1} = E00_{i0} \qquad i = 1, \ldots, N \tag{4.4}$$

$$E0_{it} = E00_{i,t-1} \qquad i = 1, \ldots, N \;\; e \;\; t = 2, \ldots, T \tag{4.5}$$

$$E00_{it} = q_{it} \qquad i = 1, \ldots, N \;\; e \;\; t = 2, \ldots, T \tag{4.6}$$

$$\sum_{i=1}^{n} (sp_i y_{iy} + b_i q_{it}) \leq C_t \qquad t = 1, \ldots, T \tag{4.7}$$

$$\sum_{i=1}^{n} w_{it} \leq 1 \qquad t = 1, \ldots, T \tag{4.8}$$

$$w_{it} \leq y_{i,t-1} + w_{i,t-1} \qquad i = 1, \ldots, N \;\; e \;\; t = 2, \ldots, T \tag{4.9}$$

$$w_{i,t+1} + w_{it} \leq 1 + z_t \qquad i = 1, \ldots, N \;\; e \;\; t = 1, \ldots, T-1 \tag{4.10}$$

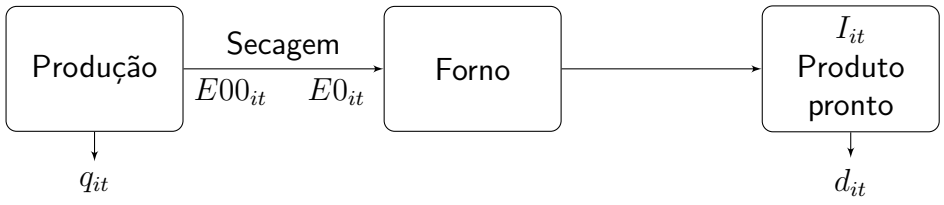
$$y_{it} + z_t \leq 1 \qquad i = 1, \ldots, N \;\; e \;\; t = 1, \ldots, T \tag{4.11}$$

$$q_{it} \leq M\,(y_{it} + w_{it}) \qquad i = 1, \ldots, N \;\; e \;\; t = 1, \ldots, T \tag{4.12}$$

$$z \in \mathbb{R}_+^T,\ q \in \mathbb{R}_+^{NT},\ I \in \mathbb{R}_+^{NT},\ E0 \in \mathbb{R}_+^{NT},\ E00 \in \mathbb{R}_+^{NT},\ y \in B^{NT} \text{ e } w \in B^{NT} \tag{4.13}$$

A Figura 27 mostra as fases do processo, desde a produção até o produto pronto. É possível verificar em que fase cada variável se encontra, sendo variável que representa uma fase inicial, intermediária ou final (do produto acabado).

Figura 27 – FASES DO PROCESSO

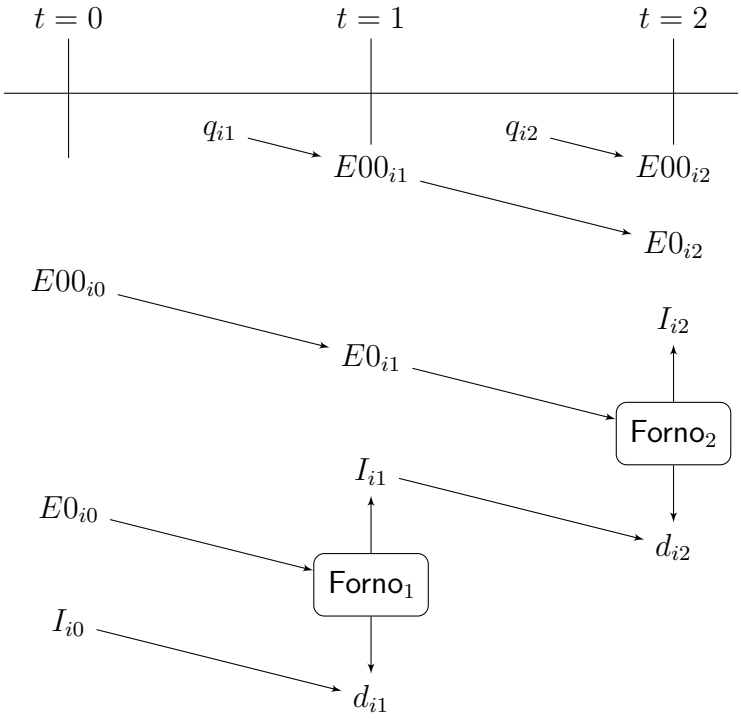


A Figura 28 representa o caminho das variáveis do modelo de um período a outro, ou seja, quais variáveis assumem os valores daquelas do período anterior (de acordo com a ordem dos processos).

Uma dificuldade nesse modelo é a etapa que consiste na queima do produto. Devido a uma espera de até duas semanas para a ida ao forno, faz-se necessária a inserção de duas variáveis a mais ($E0_{it}$ e $E00_{it}$), que representam a quantidade de produto que está em uma fase intermediária (de secagem).

Este modelo será o usado nos Testes Computacionais. As variáveis binárias y, w e z são previamente definidas por um movimento do PSO. As quantidades q, I, $E0$ e $E00$ são então encontradas por este modelo. A solução é avaliada pela expressão 4.1.

Figura 28 – VARIÁVEIS



4.4 DADOS

A indústria em questão fabrica 29 tipos de tijolos, mas possui 41 diferentes tipos de paletes (alguns com mesmo tipo de tijolo mas quantidades diferentes). Os tijolos variam em tamanho e quantidade de furos. Nesta indústria são fabricados tijolos de 6, 8, 9 e 16 furos. Além disso, ainda são fabricados Blocos Estruturais, de Vedação, Lajotas e Canaletas. Cada item possui uma largura e altura específica, que depende da forma inserida no maquinário. Para o comprimento, existem 4 tamanhos possíveis para a fabricação de todos esses itens: 9, 14, 19 ou 29 centímetros. Toda essa variação deve ser inserida no problema, visto que são vários itens a ser produzidos, cada um com uma forma e tamanho de corte muitas vezes diferente do que foi produzido anteriormente.

A tabela 6 mostra os tipos de tijolos fabricados por essa indústria, com seu respectivo tamanho e quantidade de tijolos por paletes. A sigla ABC é para representar a empresa para a qual a indústria em questão vende paletes com quantidades diferenciadas. Essa diferença deve-se ao fato de que a construtora utiliza elevadores, os quais não suportam palete de tamanho padrão. Ainda há diferença de quantidade de tijolos por palete para os meio-tijolos (que vêm acompanhados da fração $1/2$).

O tijolo de 9 furos X é um tijolo que é entregue em São Paulo e, por isso, para otimizar a viagem, carregam o caminhão com paletes mais cheios de tijolos. O palete, nesse caso, fica bem mais alto para aproveitar melhor o espaço do caminhão.

Tabela 6 – TIPOS DE CERÂMICAS FABRICADAS

TIPO DA CERÂMICA	TAMANHO (EM CM) LARG. x ALT. x COMPR.	QUANTIDADE POR PALETE
6 furos	9x14x19	500
6 furos (1/2)	9x14x09	500
6 furos (1/2)	9x14x09	1000
8 furos	9x19x19	432
8 furos (1/2)	9x19x09	432
8 furos (ABC)	9x19x19	824
8 furos (ABC 1/2)	9x19x09	600
8 furos (ABC 1/2)	9x19x09	540
8 furos	9x19x29	288
8 furos (1/2)	9x19x14	504
8 furos	11,5x19x09	348
8 furos (1/2)	11,5x19x09	348
8 furos	11,5x19x29	240
8 furos (1/2)	11,5x19x14	464
9 furos (ABC)	14x19x19	216
9 furos	14x19x19	300
9 furos (1/2)	14x19x09	360
9 furos X	14x19x24	336
9 furos	14x19x29	192
9 furos (1/2)	14x19x14	350
16 furos	19x19x19	216
16 furos (1/2)	19x19x09	216
16 furos (ABC)	19x19x19	180
16 furos (ABC)	19x19x19	216
16 furos	19x19x29	144
16 furos (1/2)	19x19x14	144
Lajota	7x30x19	324
Lajota	7x30x19	378
Bloco Vedação	9x19x29	288
Bloco Vedação (1/2)	9x19x29	288
Bloco Estrutural	14x19x29	189
Bloco Estrutural	14x19x29	164
Bloco Vedação 2º	14x19x29	192
Bloco Vedaçao 2º (1/2)	14x19x14	336
Canaleta U "19"	9x19x19	360
Canaleta U "19"	11,5x19x19	300
Canaleta U "19"	14x19x19	216
Canaleta U "19"	19x19x19	180
Canaleta U "29"	9x19x29	240
Canaleta U "29"	11,5x19x29	216
Canaleta U "29"	14x19x29	192

4.5 TESTES COMPUTACIONAIS

Os testes computacionais foram realizados por uma CPU com dois processadores IntelCore i5 de 2.27GHz, memória RAM de 6GB e Sistema Operacional de 64 bits, com o programa de linguagem computacional Matlab 7.9.0 e o otimizador Lingo 13.0 64-bits.

Para a realização dos testes simularam-se alguns dados de demanda, tempo de produção, tempo de preparação, capacidade de mão-de-obra, custo de estoque, custo de preparação e capacidade do forno. As Tabelas 7, 8 e 9 fornecem algumas dessas informações acerca do problema no qual foi aplicada a metodologia proposta. O período é de uma semana, a capacidade do forno utilizada foi de 650.000 itens e a capacidade de mão-de-obra foi de 144.000 segundos para cada período. O problema possui 1688 variáveis e 1871 restrições.

Tabela 7 – DEMANDA: ITEM (i) x PERÍODO (t)

continua

i \ t	1	2	3	4	5	6	7	8
1	25000	60000	5000	9000	10000	10000	5000	9000
2	30000	40000	21000	14000	12000	14000	21000	14000
3	10000	0	14000	17000	10000	8000	14000	17000
4	0	0	3000	9000	3000	10000	3000	9000
5	11000	20000	8000	10000	11000	12000	8000	10000
6	20000	25000	50000	9000	10000	10000	5000	11000
7	12000	0	21000	14000	12000	14000	21000	14000
8	45000	0	14000	17000	10000	8000	14000	17000
9	0	15000	0	9000	3000	10000	3000	9000
10	0	20000	0	10000	11000	12000	8000	10000
11	50000	5000	50000	9000	10000	10000	5000	9000
12	30000	35000	23000	14000	12000	14000	21000	14000
13	0	8000	0	17000	10000	8000	14000	17000
14	0	10000	0	9000	3000	10000	3000	9000
15	0	12000	0	10000	11000	12000	8000	10000
16	45000	10000	41000	9000	50000	10000	5000	10000
17	30000	14000	21000	14000	12000	14000	40000	14000
18	10000	0	0	17000	10000	18000	10000	17000
19	0	0	0	9000	3000	3000	3000	9000
20	40000	0	31000	10000	11000	24000	8000	10000
21	0	50000	50000	9000	10000	10000	5000	13000
22	24000	0	21000	14000	12000	14000	21000	14000
23	0	7000	0	17000	10000	80000	31000	30000
24	0	0	0	9000	40000	10000	35000	15000

Tabela 7 – DEMANDA: ITEM (i) x PERÍODO (t)

conclusão

i \ t	1	2	3	4	5	6	7	8
25	22000	36000	0	10000	11000	12000	36000	15000
26	20000	25000	39000	0	20000	10000	50000	9000
27	25000	14000	0	14000	12000	16000	21000	14000
28	0	0	40000	17000	50000	8000	14000	31000
29	0	10000	0	9000	3000	1500	30000	18000
30	11000	12000	80000	10000	5000	10000	8000	25000
31	30000	0	21000	9000	10000	10000	5000	17000
32	24000	50000	0	14000	35000	10000	45000	14000
33	60000	0	50000	17000	10000	8000	14000	17000
34	0	25000	0	9000	3000	10000	3000	7000
35	0	10000	0	10000	11000	12000	8000	10000

Tabela 8 – ESTOQUES INICIAIS

continua

ITEM	E0	EI0	EI00
1	0	25000	60000
2	0	30000	40000
3	0	10000	0
4	0	0	0
5	0	11000	20000
6	0	20000	25000
7	0	12000	0
8	0	45000	0
9	0	0	15000
10	0	0	20000
11	0	50000	5000
12	0	30000	35000
13	0	0	8000
14	0	0	10000
15	0	0	12000
16	0	45000	10000
17	0	30000	14000
18	0	10000	0
19	0	0	0
20	0	40000	0

Tabela 8 – ESTOQUES INICIAIS

conclusão

ITEM	E0	EI0	EI00
21	0	0	50000
22	0	24000	0
23	0	0	7000
24	0	0	0
25	0	22000	36000
26	0	20000	25000
27	0	25000	14000
28	0	0	0
29	0	0	10000
30	0	11000	12000
31	0	30000	0
32	0	24000	50000
33	0	60000	0
34	0	0	25000
35	0	0	10000

Tabela 9 – DADOS (POR ITEM)

continua

ITEM	b	sp	h	s
1	0,2	15	0,15	0,35
2	0,3	10	0,1	0,2
3	0,2	25	0,12	0,25
4	0,5	30	0,1	0,1
5	0,2	8	0,13	0,15
6	0,2	15	0,15	0,35
7	0,3	10	0,1	0,9
8	0,3	25	0,12	0,41
9	0,4	30	0,1	0,11
10	0,3	8	0,13	0,22
11	0,2	15	0,15	0,32
12	0,3	10	0,1	0,15
13	0,4	25	0,12	0,75
14	0,5	30	0,1	0,12
15	0,3	8	0,13	0,15
16	0,2	15	0,15	0,12
17	0,3	10	0,1	0,35

Tabela 9 – DADOS (POR ITEM)
conclusão

ITEM	b	sp	h	s
18	0,4	25	0,12	0,2
19	0,5	30	0,1	0,25
20	0,3	8	0,13	0,1
21	0,2	15	0,15	0,15
22	0,2	10	0,1	0,35
23	0,4	25	0,12	0,9
24	0,5	30	0,1	0,41
25	0,3	8	0,13	0,11
26	0,2	15	0,15	0,22
27	0,3	10	0,1	0,32
28	0,2	25	0,12	0,15
29	0,5	30	0,1	0,75
30	0,3	8	0,13	0,12
31	0,2	15	0,15	0,15
32	0,3	10	0,1	0,12
33	0,2	25	0,12	0,25
34	0,4	30	0,1	0,1
35	0,3	8	0,13	0,15

Os resultados obtidos, tempo computacional e tamanho do problema (número de itens e períodos) estão descritos na Tabela 10. A função objetivo refere-se a melhor solução encontrada dentro daquele tempo computacional.

Tabela 10 – TESTES COMPUTACIONAIS

ITENS x PERÍODOS	FUNÇÃO OBJETIVO	PROGRAMA COMPUTACIONAL	TEMPO (EM SEGUNDOS)
35x8	28253,5	Lingo	3600

Pelo PSO programado no Matlab não foi melhorada nenhuma solução inicial e, portanto, não teve um resultado a ser inserido na tabela. Para um problema deste tamanho a programação do PSO demorou para alterar os resultados. Cada iteração foi completada em 72 segundos. O programa demorou para encontrar uma solução factível diferente das soluções iniciais, o que fica totalmente inviável e o que não aconteceu em até 1000 iterações (20 horas aproximadamente).

A cada movimentação é feita a distribuição de quantidades pelo Lingo e essa distribuição que acarreta no tempo computacional muito elevado, visto que é feita a chamada do

Lingo inúmeras vezes. Sendo assim, a aplicação proposta, da maneira que foi codificada, não foi eficiente, pois nem o tempo computacional nem a função objetivo foram melhorados.

O Lingo obteve o resultado dado pela Tabela 11. Pode-se perceber que:

- Nos dois primeiros períodos foi utilizado tijolo dos estoques iniciais para atender a demanda;
- Em alguns períodos foi produzido mais do que a demanda do período e mantido o restante em estoque final, como por exemplo a produção do 1º item no 4º período (cuja demanda a ser atendida é a do 6º período e o restante mantido em estoque para atender a demanda dos períodos seguintes).
- Nos dois últimos períodos não há produção para que não haja estoque final.

Tabela 11 – RESULTADO ENCONTRADO PELO LINGO:
ITEM (i) x PERÍODO (t)

continua

i \ t	1	2	3	4	5	6	7	8
1	5000	9000	10000	15000	0	9000	0	0
2	21000	14000	12000	14000	21000	14000	0	0
3	14000	17000	18000	0	14000	17000	0	0
4	5180	9820	0	14838	0	7162	0	0
5	8000	10000	11000	12000	8000	10000	0	0
6	50000	9000	10000	15000	0	11000	0	0
7	21000	14000	26000	0	21000	14000	0	0
8	14000	17000	18000	0	14000	17000	0	0
9	0	12000	0	13000	0	9000	0	0
10	0	10000	11000	12000	8000	10000	0	0
11	50000	9000	10000	15000	0	9000	0	0
12	23000	14000	12000	14000	21000	14000	0	0
13	0	17000	32000	0	0	17000	0	0
14	0	12000	0	22000	0	0	0	0
15	0	10000	11000	12000	8000	10000	0	0
16	41000	9000	50000	15000	0	10000	0	0
17	21000	14000	12000	14000	40000	14000	0	0
18	0	27000	0	28000	0	17000	0	0
19	0	18000	0	0	0	9000	0	0
20	31000	10000	11000	24000	8000	10000	0	0
21	50000	9000	10000	15000	0	13000	0	0

Tabela 11 – RESULTADO ENCONTRADO PELO LINGO:
ITEM (i) x PERÍODO (t)

conclusão

i \ t	1	2	3	4	5	6	7	8
22	21000	14000	12000	14000	21000	14000	0	0
23	0	27000	0	80000	31000	30000	0	0
24	0	9338	51888	0	32774	15000	0	0
25	0	10000	11000	12000	36000	15000	0	0
26	39000	0	20000	10000	50000	9000	0	0
27	0	14000	12000	16000	21000	14000	0	0
28	40000	17000	58000	0	14000	31000	0	0
29	0	13500	0	0	31674	16326	0	0
30	80000	10000	5000	10000	8000	25000	0	0
31	21000	9000	10000	15000	0	17000	0	0
32	0	14000	45000	0	45000	14000	0	0
33	50000	17000	10000	22000	0	17000	0	0
34	0	12000	0	13000	0	7000	0	0
35	0	10000	11000	12000	8000	10000	0	0

Devido ao elevado tempo computacional não foi encontrada, através do PSO, alguma solução diferente da inicial. Não foi satisfatória essa aplicação do PSO no problema de Planejamento de Produção de modo que possa ser aproveitada pela indústria.

5 CONCLUSÃO

O objetivo geral do trabalho era verificar a aplicabilidade da metaheurística PSO e uma metodologia exata em um problema de uma indústria. Foram estudados, primeiramente, os problemas de produção para compreender as diferenças entre os diversos problemas existentes. Com uma revisão bibliográfica sobre alguns problemas de planejamento de produção e também uma seleção de alguns artigos que aplicam o PSO em problemas de produção, foi possível verificar que a aplicação do PSO discreto em um problema de planejamento de produção não é facilmente encontrado. Existem vastas aplicações do PSO em problemas de sequenciamento, como *Flow Shop* e *Job Shop* e também aplicações de outras metaheurísticas, como Algoritmo Genético, Colônia de Formigas e Busca Tabu, em problemas de planejamento, mas a pouca aplicação do PSO em problemas de planejamento de produção foi o que motivou o tema deste trabalho.

Depois de determinado que seria estudado o PSO em um problema de planejamento de produção, foi necessário o estudo deste problema específico e também da metaheurística, sendo essa aplicada a matrizes com valores contínuos, discretos ou binários. Realizado o estudo sobre a metaheurística a ser aplicada e o problema de produção escolhido, fez-se necessário estudar a forma de aplicabilidade do PSO no problema de produção da indústria de tijolos.

Na aplicação foi necessário o uso de variáveis discretas para representar as decisões a serem tomadas e, portanto, foi utilizada uma metodologia exata, através do otimizador, para a distribuição das quantidades e cálculo do valor da função objetivo. Desenvolveu-se então um programa em uma linguagem computacional que fizesse a movimentação da partícula da nuvem e, em conjunto com a metodologia exata, retornasse o valor da função objetivo.

Alguns desafios que geraram um alto custo computacional na aplicação desta proposta merecem mais discussão: codificação dos elementos do problema discreto, representação das soluções, tratamento das infactibilidades na solução inicial e nas soluções encontradas pela movimentação da nuvem e diversificação das soluções iniciais.

Através da análise dos testes computacionais, foi percebido que a aplicação proposta de movimentação de uma matriz discreta pelo PSO em um problema de planejamento de produção da indústria de tijolos não foi eficiente. Notou-se a grande dificuldade em se trabalhar com o PSO discreto, mas trabalhar com um PSO contínuo, movimentando a matriz de quantidades, foi inviável devido à natureza do problema. Uma das grandes dificuldades em se trabalhar com o PSO discreto foram as infactibilidades encontradas no processo. Quando a matriz era movimentada, muitas vezes a solução encontrada era infactível e, portanto, uma nova iteração era necessária.

A chamada do otimizador é o que levou ao grande tempo computacional. O grande

número de infactibilidades junto com o grande tempo computacional aumentam a dificuldade em se encontrar a solução ótima, por causa da demora de cada iteração. Como o problema da indústria de tijolos é um problema de grande porte, o PSO precisaria de muito mais iterações para que tivesse um resultado melhor, o que resultaria em um tempo computacional muito mais elevado. Portanto, é necessária uma mudança na programação em relação à matriz a ser movimentada ou também quanto à chamada do otimizador pelo programa computacional.

A representação das matrizes foi outra grande dificuldade pois também contribuiu para as infactibilidades encontradas. Um melhor tratamento dessas infactibilidades faz-se necessário, tanto para a geração das soluções iniciais quanto para as soluções encontradas com a movimentação da nuvem. As soluções iniciais foram encontradas por heurísticas que também podem ser diversificadas, ampliando o espaço de busca, o que facilita a determinação da função objetivo mínima sem cair em mínimos locais.

Conclui-se que a aplicabilidade do PSO combinado com uma metodologia exata em um problema de planejamento de produção de uma indústria de tijolos não foi eficiente da maneira proposta. Para estudos futuros uma sugestão é estudar a mudança de aplicação, que pode ser a aplicação somente do PSO (sem a metodologia exata) ou também uma mudança na matriz a ser movimentada pelo PSO. Algumas outras sugestões para trabalhos futuros são: mudar o *software* (utilizar outros *softwares* como *CPlex*) ou alterar a forma de chamada do Lingo no Matlab, utilizando a função *linprog*; reduzir o espaço de busca evitando as infactibilidades; mudar o horizonte de planejamento para um período maior; mudar a representação das soluções e tentar uma adaptação para um PSO contínuo.

REFERÊNCIAS

ARENALES, M. et al. **Pesquisa Operacional**. 6ª edição. ed. [S.l.]: Campus - ABEPRO, 2007. Citado 3 vezes nas páginas 25, 26 e 47.

BANK, M. et al. Application of particle swarm optimization and simulated annealing algorithms in flow shop scheduling problem under linear deterioration. **Advances in Engineering Software**, 2012. Citado na página 37.

CHAKRABORTTY, R. K. et al. A possibilistic environment based particle swarm optimization for aggregate production planning. **Computers & Industrial Engineering**, 2015. Citado na página 36.

FLEISCHMANN, B.; MEYR, H. The general lotsizing and scheduling problem. **OR Spektrum**, 1997. Citado 3 vezes nas páginas 36, 39 e 41.

JARBOUI, B. et al. A combinatorial particle swarm optimization for solving permutation flowshop problems. **Computers & Industrial Engineering**, 2008. Citado na página 37.

KASHAN, A. H.; KARIMI, B. A discrete particle swarm optimization algorithm for scheduling parallel machines. **Computers & Industrial Engineering**, 2009. Citado na página 35.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. **IEEE International Conference on Neural Networks**, p. 1942–1948, 1995. Citado 2 vezes nas páginas 31 e 34.

KENNEDY, J.; EBERHART, R. C. A discrete binary version of the particle swarm algorithm. **IEEE International Conference on Computational Cybernetics and Simulation**, 1997. Citado 2 vezes nas páginas 35 e 36.

KENNEDY, J.; EBERHART, R. C. **Swarm Intelligence**. [S.l.]: Morgan Kaufmann Publishers, 2001. Citado na página 35.

KULKARNI, N. K. et al. Particle swarm optimization applications to mechanical engineering - a review. **Material Today: Proceedings**, n. 2, p. 2631–2639, 2015. Citado na página 36.

LIAO, C.-J.; TSENG, C.-T.; LUARN, P. A discrete version of particle swarm optimization for flowshop scheduling problems. **Computers & Operations Research**, 2007. Citado na página 36.

LIU, B.; WANG, L.; JIN, Y.-H. An effective hybrid pso-based algorithm for flow shop scheduling with limited buffers. **Computers & Operations Research**, 2008. Citado na página 37.

MARINI, F.; WALCZAK, B. Particle swarm optimization (pso). a tutorial. **Chemometrics and Intelligent Laborary Systems**, n. 149, p. 153–165, 2015. Citado na página 32.

OBAL, T. M. **Desenvolvimento e avaliação de Matheurísticas para o combinado problema do posicionamento dos feixes e distribuição de dose no planejamento de radioterapia**. Tese (Doutorado) — Universidade Federal do Paraná, 2016. Citado na página 37.

PAN, Q.-K.; TASGETIREN, M. F.; LIANG, Y.-C. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. **Computers & Operations Research**, 2008. Citado na página 35.

SHA, D.; LIN, H.-H. A multi-objective pso for job-shop scheduling problems. **Expert Systems with Applications**, 2010. Citado na página 37.

TASGETIREN, M. F.; LIANG, Y.-C. A binary particle swarm optimization algorithm for lot sizing problem. **Journal of Economic and Social Research**, 2003. Citado 2 vezes nas páginas 35 e 36.

TOLEDO, C. F. M. et al. Metaheurísticas aplicadas ao problema geral de dimensionamento de lotes e programação da produção. **XLI SBPO**, 2009. Citado na página 36.

TOLEDO, C. F. M.; FRANÇA, P. M.; MORABITO, R. Resolução de um problema integrado de dimensionamento de lotes e programação da produção utilizando algoritmo genético multi-populacional. **XXVI ENEGEP**, 2006. Citado na página 36.

WANG, S.-C.; YEH, M.-F. A modified particle swarm optimization for aggregate production planning. **Expert Systems with Applications**, 2014. Citado na página 36.

ZHAO, F. et al. An improved particle swarm optimization with decline disturbance index (ddpso) for multi-objective job-shop scheduling problem. **Computers & Operations Research**, 2014. Citado na página 37.

APÊNDICES

O modelo abaixo foi programado no Lingo para o problema da Seção 3.4.

```
MODEL:

DATA:
J=4; !numero de periodos;
n=3; !numero de itens;
ENDDATA

SETS:
periodo / 1..J /: capacidade, q; !indice t;
item / 1..n /: custopreparacao, custoestoque, estoque0, tempopreparo,
    tempoproducao; !indice i!;
matriz(item,periodo): x, y, estoque, demanda, w; !, M;
ENDSETS

DATA:

custopreparacao = 350, 100, 90;
custoestoque = 150, 100, 70;
estoque0 = 0, 0, 0;
tempopreparo = 40, 40, 40;
tempoproducao = 20, 10, 20;
capacidade = 280, 320, 280, 400;
demanda = 1,10,3,10,2,4,0,5,2,4,0,5;


ENDDATA

!FO;
[FO] MIN=@SUM(matriz(i,t): custopreparacao(i)*y(i,t)+custoestoque(i)*
    estoque(i,t));

!Balanceamento de Estoque;
@FOR(matriz(i,t)| t#EQ#1: estoque(i,t)=estoque0(i)+x(i,t)-
    demanda(i,t));
@FOR(matriz(i,t)| t#GE#2: estoque(i,t)=estoque(i,t-1)+x(i,t)-
    demanda(i,t));
```

```
@FOR(periodo(t): @SUM(item(i):(tempopreparo(i)*y(i,t)+tempoproducao(i)*
    x(i,t)))<=capacidade(t));

@FOR(periodo(t): @SUM(item(i):w(i,t))<=1);

@FOR(periodo(t)| t#GE#2: @FOR(item(i): w(i,t) <= y(i,t-1)+w(i,t-1)));

@FOR(periodo(t)| J#GT#t: @FOR(item(i): w(i,t+1) + w(i,t) <= 1+ q(t)));

@FOR(periodo(t): @FOR(item(i): y(i,t)+q(t) <= 1 ));

@FOR(periodo(t): @FOR(item(i): x(i,t) <= 10000* ( y(i,t)+w(i,t) ) ));

@FOR(periodo(t) | (J#GT#t) #AND# (t#NE#1): @BIN(q(t)));
@FOR(periodo(t) | (t#EQ#1) #OR# (t#EQ#J): q(t)=0);

@FOR(periodo(t) | (t#NE#1): @FOR(item(i): @BIN(w(i,t))));
@FOR(periodo(t) | (t#EQ#1): @FOR(item(i): w(i,t)=0));

@FOR(periodo(t): @FOR(item(i): @BIN(y(i,t))));

DATA:
   @text('Output.txt') = FO;
ENDDATA

END

GO
QUIT
```

Segue abaixo o modelo programado para o problema da Seção 4.5.

```
MODEL:

DATA:
J=8; !numero de periodos;
n=35; !numero de itens;
capacidadeforno = 650000;
ENDDATA

SETS:
periodo / 1..J /: capacidade, q; !indice t;
item / 1..n/: custopreparacao, custoestoque, estoque0, tempopreparo,
    tempoproducao, estoqueint0inicial, estoqueint00inicial; !indice i!;
matriz(item,periodo): estoque, demanda, x, y, w, M, estoqueint0,
    estoqueint00;

ENDSETS

DATA:
w = @FIlE('Input.txt');
y = @FILE('Input.txt');
q = @FILE('Input.txt');
demanda, custopreparacao, custoestoque, capacidade, tempoproducao,
    tempopreparo, estoque0, estoqueint0inicial, estoqueint00inicial =
    @OLE('Olaria1.xls');

ENDDATA

!FO;
[FO] MIN=@SUM(matriz(i,t): custopreparacao(i)*y(i,t)+custoestoque(i)*
    estoque(i,t));!+custoestoqueI(i)*estoqueintermediario(i,t));
!Balanceamento de Estoque pronto;
@FOR(matriz(i,t)| t#EQ#1: estoque(i,t)=estoque0(i)+estoqueint0inicial(i)-
    demanda(i,t));
@FOR(matriz(i,t)| t#GE#2: estoque(i,t)=estoque(i,t-1)+estoqueint0(i,t-1)-
    demanda(i,t));
@FOR(matriz(i,t)| t#EQ#1: estoqueint0(i,t)=estoqueint00inicial(i));
@FOR(matriz(i,t)| t#GE#2: estoqueint0(i,t)=estoqueint00(i,t-1));
```

```
@FOR(matriz(i,t)| t#GE#1: estoqueint00(i,t)=x(i,t));
@FOR(periodo(t): @SUM(item(i):(tempopreparo(i)*y(i,t)+tempoproducao(i)*
    x(i,t)))<=capacidade(t));
@FOR(periodo(t): @SUM(item(i):w(i,t))<=1);
@FOR(periodo(t)| t#GE#2: @FOR(item(i): w(i,t) <= y(i,t-1)+w(i,t-1) ));
@FOR(periodo(t)| J#GT#t: @FOR(item(i): w(i,t+1) + w(i,t) <= 1+ q(t)));
@FOR(periodo(t): @FOR(item(i): y(i,t)+q(t) <= 1 ));
@FOR(periodo(t): @FOR(item(i): x(i,t) <= M(i,t)* ( y(i,t)+w(i,t) ) ));
@FOR(periodo(t): @FOR(item(i):M(i,t)=@SMIN((capacidade(t)-
    tempopreparo(i))/tempoproducao(i) , @SUM(periodo(R)| (R#GE#t) #AND#
    (R#LE#J):demanda(i,R)) ) ));
@FOR(periodo(t): @SUM(item(i): x(i,t)) <= capacidadeforno);
@FOR(periodo(t) | (J#GT#t) #AND# (t#NE#1): @BIN(q(t)));
@FOR(periodo(t) | (t#EQ#1) #OR# (t#EQ#J): q(t)=0);
@FOR(periodo(t) | (t#NE#1): @FOR(item(i): @BIN(w(i,t))));
@FOR(periodo(t) | (t#EQ#1): @FOR(item(i): w(i,t)=0));
@FOR(periodo(t): @FOR(item(i): @BIN(y(i,t))));

DATA:
@TEXT('Output.txt') = FO;
ENDDATA

END

GO
QUIT
```

Abaixo seguem as funções utilizadas na programação do Matlab.

```
function x = ProdAnteriores(num_itens,swarm,periodo,individuo)
    x(1:num_itens,1) = 0;
for t = 1:periodo
        for i = 1:num_itens
            x(i,1) = x(i,1) + swarm((t-1)*num_itens+i,individuo,2);
        end
    end
end

function x = DemNaoAtendAnteriores(num_itens,D,periodo)
x(1:num_itens,1) = 0;
for i = 1:num_itens
x(i,1) = x(i,1) + sum(D(i,3:periodo+1,1));
    end
end

function x = DemNaoAtendPosteriores(num_itens,D,periodo,num_periodos)
    x(1:num_itens,1) = 0;
for i = 1:num_itens
        x(i,1) = x(i,1) + sum(D(i,periodo+2:num_periodos+2,1));
    end
end

function x =
    DemNaoAtendBackward(num_itens,num_periodos,periodo,swarm,D,individuo)
    aux1(1:num_itens,1) = 0;
    aux2(1:num_itens,1) = 0;
    for i = 1:num_itens
       for t = periodo:num_periodos-2
           aux1(i,1) = aux1(i,1) + D(i,t+2,1);
           aux2(i,1) = aux2(i,1) + swarm((t-1)*num_itens+i,individuo,2);
       end
    end
    x = aux1-aux2;
end

function x = DemNaoAtendForward(num_itens,num_periodos,swarm,D,individuo)
```

```
    aux1(1:num_itens,1) = 0;
    aux2(1:num_itens,1) = 0;
    for i = 1:num_itens
       for t = 1:num_periodos-2
           aux1(i,1) = aux1(i,1) + D(i,t+2,1);
           aux2(i,1) = aux2(i,1) + swarm((t-1)*num_itens+i,individuo,2);
       end
    end
    x = aux1-aux2;
end

function x = DemVariavel(num_itens,num_periodos,D)
x(1:num_itens,1) = 0;
for i = 1:num_itens
        for t = 1:num_periodos-2
x(i,1) = x(i,1) + D(i,t+2,1);
        end
end
end

function x = Estoque(num_itens,num_periodos,swarm,individuo,D,E)

    x = E;
    aux1 = 0;
    aux2 = 0;

    for i = 1:num_itens
        for t = 1:num_periodos-2
            aux1 = aux1 + swarm((t-1)*num_itens+i,individuo,2);
            aux2 = aux2 + D(i,t+2,1);
            if aux1-aux2>0
               x((t-1)*num_itens+i,individuo,1) = aux1-aux2;
            end
        end
        aux1=0;
        aux2=0;
    end
```

```
end
```

Alguns outros programas foram criados para serem incluídos no programa principal do Matlab.

```
%CapacidadeFornoRestante

for t = 1:num_periodos
    for i = 1:num_itens
        if swarm((t-1)*num_itens+i,p,2)>0
            capacidadeforno = capacidadeforno - swarm((t-1)*
                num_itens+i,p,2);
        end
    end
end

%IndividuosBackwardOlaria
%Programa para criar os indivíduos 1, 3 e 5 da população inicial
%---------------------------------------------------------------
%primeiro e terceiro indivíduo da população

CVariavel1 = C;
CVariavel3 = C;
CVariavel5 = C;
DVar1 = DemVariavel(num_itens,num_periodos,D);
DVar3 = DemVariavel(num_itens,num_periodos,D);
DVar5 = DemVariavel(num_itens,num_periodos,D);


for t = num_periodos-2:-1:1
EstProd1 = h./b;
    EstProd3 = h./b;
    RelacaoDem = DemNaoAtendBackward(num_itens,num_periodos,t,
        swarm,D,5)./DVar5;
for i = 1:num_itens
[maxEstProd1,location1]=max(EstProd1);
        [minEstProd3,location3]=min(EstProd3);
        [maxRelacaoDem,location5]=max(RelacaoDem);
Dem1 = DVar1 - DemNaoAtendAnteriores(num_itens,D,t);
        Dem3 = DVar3 - DemNaoAtendAnteriores(num_itens,D,t);
        Dem5 = DVar5 - DemNaoAtendAnteriores(num_itens,D,t);
```

```
        if Dem1(location1) > 0 && (CVariavel1(t)-
            sp(location1))/b(location1)>0 && CapacidadeForno(1,t+2)>0;
            CVariavel1(t) = CVariavel1(t) - sp(location1);
            CapProd1 = CVariavel1(t)/b(location1);
diff = min(Dem1(location1), CapProd1);
            diff1 = min(diff,CapacidadeForno(1,t+2));
swarm((t-1)*num_itens+location1,1,2) = swarm((t-1)*
    num_itens+location1,1,2) + diff1;
CVariavel1(t) = CVariavel1(t) - diff1*b(location1);
            CapacidadeForno(1,t+2) = CapacidadeForno(1,t+2) - diff1;
DVar1(location1) = DVar1(location1) - diff1;
EstProd1(location1) = -1;
elseif (Dem1(location1)<=0 || (CVariavel1(t)-
    sp(location1))/b(location1)<=0) && swarm((t-1)*
        num_itens+location1,1,2)==0;
EstProd1(location1) = -1;
swarm((t-1)*num_itens+location1,1,1) = -1;
        end
        if Dem3(location3) > 0 && (CVariavel3(t)-
            sp(location3))/b(location3)>0 && CapacidadeForno(3,t+2)>0
CVariavel3(t) = CVariavel3(t) - sp(location3);
            CapProd3 = CVariavel3(t)/b(location3);
diff = min(Dem3(location3), CapProd3);
            diff3 = min(diff,CapacidadeForno(3,t+2));
swarm((t-1)*num_itens+location3,3,2) = swarm((t-1)*
    num_itens+location3,3,2) + diff3;
CVariavel3(t) = CVariavel3(t) - diff3*b(location3);
            CapacidadeForno(3,t+2) = CapacidadeForno(3,t+2) - diff3;
DVar3(location3) = DVar3(location3) - diff3;
EstProd3(location3) = max(EstProd3) + 1;
elseif (Dem3(location3)<=0 || (CVariavel3(t)-
    sp(location3))/b(location3)<=0) && swarm((t-1)*
        num_itens+location3,3,2)==0;
EstProd3(location3) = max(EstProd3) + 1;
swarm((t-1)*num_itens+location3,3,1) = -1;
        end
        if Dem5(location5) > 0 && (CVariavel5(t)-
            sp(location5))/b(location5)>0 && CapacidadeForno(5,t+2)>0
            CVariavel5(t) = CVariavel5(t) - sp(location5);
```

```
            CapProd5 = CVariavel5(t)/b(location5);
diff = min(Dem5(location5), CapProd5);
            diff5 = min(diff,CapacidadeForno(5,t+2));
swarm((t-1)*num_itens+location5,5,2) = swarm((t-1)*
    num_itens+location5,5,2) + diff5;
CVariavel5(t) = CVariavel5(t) - diff5*b(location5);
            CapacidadeForno(5,t+2) = CapacidadeForno(5,t+2) - diff5;
DVar5(location5) = DVar5(location5) - diff5;
RelacaoDem(location5) = -1;
elseif (Dem5(location5)<=0 || (CVariavel5(t)-
    sp(location5))/b(location5)<=0) && swarm((t-1)*
        num_itens+location5,5,2)==0;
RelacaoDem(location5) = -1;
swarm((t-1)*num_itens+location5,5,1) = -1;
        end
    end
end

DVar5 = DemVariavel(num_itens,num_periodos,D);

for t = 2:num_periodos-2
    EstProd1 = h./b;
    EstProd3 = h./b;
    RelacaoDem = 1./DVar5;
    AuxPrep1 = 0;
    AuxPrep3 = 0;
    AuxPrep5 = 0;
    for i = 1:num_itens
        [maxEstProd1,location1]=max(EstProd1);
        [minEstProd3,location3]=min(EstProd3);
        [maxRelacaoDem,location5]=max(RelacaoDem);
        if AuxPrep1==0 && swarm((t-1)*num_itens+location1,1,1)==1 &&
        swarm((t-2)*num_itens+location1,1,1)==1
            swarm((t-1)*num_itens+location1,1,1) = 0;
            AuxPrep1=1;
        end
        if AuxPrep3==0 && swarm((t-1)*num_itens+location3,3,1)==1 &&
        swarm((t-2)*num_itens+location3,3,1)==1
            swarm((t-1)*num_itens+location3,3,1) = 0;
```

```
            AuxPrep3=1;
        end
        if AuxPrep5==0 && swarm((t-1)*num_itens+location5,5,1)==1 &&
        swarm((t-2)*num_itens+location5,5,1)==1
            swarm((t-1)*num_itens+location5,5,1) = 0;
            AuxPrep5=1;
        end
        EstProd1(location1) = -1;
        EstProd3(location3) = max(EstProd3) + 1;
        RelacaoDem(location5) = -1;
    end
end

clear CVariavel1 & CVariavel3 & EstProd1 & EstProd3 & Dem1 & Dem3;
clear location1 & location3 & DVar1 & DVar3 & diff1 & diff3;
clear CVariavel5 & RelacaoDem & Dem5 & location5 & DVar5 & diff5;
clear maxEstProd1 & maxEstProd3 & minEstProd3 & CapProd1 & CapProd3;
clear CapProd5 & maxRelacaoDem;
clear AuxPrep1 & AuxPrep3 & AuxPrep5;


%IndividuosForwardOlaria
%-----------------------------------------------------------------
%segundo, quarto e sexto indivíduo da população

CVariavel2 = C;
CVariavel4 = C;
CVariavel6 = C;
DVar2 = DemVariavel(num_itens,num_periodos,D);
DVar4 = DemVariavel(num_itens,num_periodos,D);
DVar6 = DemVariavel(num_itens,num_periodos,D);


for t = 1:num_periodos-2
EstProd2 = h./b;
    EstProd4 = h./b;
    RelacaoDem = DemNaoAtendForward(num_itens,num_periodos,swarm,D,6);
    AuxPrep2 = 0;
    AuxPrep4 = 0;
```

```
    AuxPrep6 = 0;
for i = 1:num_itens
[minEstProd2,location2]=min(EstProd2);
        [maxEstProd4,location4]=max(EstProd4);
        [minRelacaoDem,location6]=min(RelacaoDem);
        x=DemNaoAtendAnteriores(num_itens,D,t);
        y2=ProdAnteriores(num_itens,swarm,t,2);
        y4=ProdAnteriores(num_itens,swarm,t,4);
        y6=ProdAnteriores(num_itens,swarm,t,6);
Dem2 = D(location2,t+2,1)+x(location2)-y2(location2);
        Dem4 = D(location4,t+2,1)+x(location4)-y4(location4);
        Dem6 = D(location6,t+2,1)+x(location6)-y6(location6);
if Dem2 > 0 && (CVariavel2(t)-
    sp(location2))/b(location2)>0 && CapacidadeForno(2,t+2)>0;
if t==1 || swarm((t-2)*num_itens+location2,2,1)~=1 ||
    AuxPrep2==1
                CVariavel2(t) = CVariavel2(t) - sp(location2);
                swarm((t-1)*num_itens+location2,2,1)=1;
            else
                AuxPrep2 = 1;
                swarm((t-1)*num_itens+location2,2,1)=0;
            end
CapProd2 = CVariavel2(t)/b(location2);
diff = min(Dem2, CapProd2);
            diff2 = min(diff,CapacidadeForno(2,t+2));
swarm((t-1)*num_itens+location2,2,2) = swarm((t-1)*
    num_itens+location2,2,2) + diff2;
CVariavel2(t) = CVariavel2(t) - diff2*b(location2);
            CapacidadeForno(2,t+2) = CapacidadeForno(2,t+2) - diff2;
DVar2(location2) = DVar2(location2) - diff2;
EstProd2(location2) = max(EstProd2) + 1;
elseif Dem2<=0 || CVariavel2(t)/b(location2)==0;
EstProd2(location2) = max(EstProd2) + 1;
swarm((t-1)*num_itens+location2,2,1) = -1;
        end
if Dem4 > 0 && (CVariavel4(t)-
    sp(location4))/b(location4)>0 && CapacidadeForno(4,t+2)>0 ;
            if t==1 || swarm((t-2)*num_itens+location4,4,1)~=1 ||
                AuxPrep4==1
```

```
                CVariavel4(t) = CVariavel4(t) - sp(location4);
                swarm((t-1)*num_itens+location4,4,1) = 1;
            else
                AuxPrep4 = 1;
                swarm((t-1)*num_itens+location4,4,1)=0;
            end
CapProd4 = CVariavel4(t)/b(location4);
diff = min(Dem4, CapProd4);
            diff4 = min(diff,CapacidadeForno(4,t+2));
swarm((t-1)*num_itens+location4,4,2) = swarm((t-1)*
    num_itens+location4,4,2) + diff4;
CVariavel4(t) = CVariavel4(t) - diff4*b(location4);
            CapacidadeForno(4,t+2) = CapacidadeForno(4,t+2) - diff4;
DVar4(location4) = DVar4(location4) - diff4;
EstProd4(location4) = -1;
elseif Dem4<=0 || CVariavel4(t)/b(location4)==0;
EstProd4(location4) = -1;
swarm((t-1)*num_itens+location4,4,1) = -1;
        end
        if Dem6 > 0 && (CVariavel6(t)-
            sp(location6))/b(location6)>0 && CapacidadeForno(6,t+2)>0;
            if t==1 || swarm((t-2)*num_itens+location6,6,1)~=1 ||
                AuxPrep6==1
                CVariavel6(t) = CVariavel6(t) - sp(location6);
                swarm((t-1)*num_itens+location6,6,1) = 1;
            else
                AuxPrep6 = 1;
                swarm((t-1)*num_itens+location6,6,1)=0;
            end
CapProd6 = CVariavel6(t)/b(location6);
diff = min(Dem6, CapProd6);
            diff6 = min(diff,CapacidadeForno(6,t+2));
swarm((t-1)*num_itens+location6,6,2) = swarm((t-1)*
    num_itens+location6,6,2) + diff6;
CVariavel6(t) = CVariavel6(t) - diff6*b(location6);
            CapacidadeForno(6,t+2) = CapacidadeForno(6,t+2) - diff6;
DVar6(location6) = DVar6(location6) - diff6;
RelacaoDem(location6) = max(RelacaoDem) + 1;
elseif Dem6<=0 || CVariavel6(t)/b(location6)==0;
```

```
RelacaoDem(location6) = max(RelacaoDem) + 1;
swarm((t-1)*num_itens+location6,6,1) = -1;
        end
    end

    aux = t+1;
    while (CVariavel2(t)>0 || CVariavel4(t)>0 || CVariavel6(t)>0)
        && aux < num_periodos+1 && t<num_periodos-2
        EstProd2 = h./b;
        EstProd4 = h./b;
        RelacaoDem = DemNaoAtendForward(num_itens,num_periodos,swarm,D,6);
        for i=1:num_itens
            [minEstProd2,location2]=min(EstProd2);
            [maxEstProd4,location4]=max(EstProd4);
            [minRelacaoDem,location6]=min(RelacaoDem);
            if CapacidadeForno(2,t+2)>0 && CVariavel2(t)>0 &&
                swarm((t-1)*num_itens+location2,2,1)~=-1;
                CapProd2 = CVariavel2(t)/b(location2);
                diff = min(D(location2,aux,1), CapProd2);
                diff2 = min(diff,CapacidadeForno(2,t+2));
                swarm((t-1)*num_itens+location2,2,2) =
                    swarm((t-1)*num_itens+location2,2,2) + diff2;
                CVariavel2(t) = CVariavel2(t) - diff2*b(location2);
                CapacidadeForno(2,t+2) = CapacidadeForno(2,t+2) - diff2;
                DVar2(location2) = DVar2(location2) - diff2;
            end
            if CapacidadeForno(4,t+2)>0 && CVariavel4(t)>0 &&
                swarm((t-1)*num_itens+location4,4,1)~=-1;
                CapProd4 = CVariavel4(t)/b(location4);
                diff = min(D(location4,aux,1), CapProd4);
                diff4 = min(diff,CapacidadeForno(4,t+2));
                swarm((t-1)*num_itens+location4,4,2) =
                    swarm((t-1)*num_itens+location4,4,2) + diff4;
                CVariavel4(t) = CVariavel4(t) - diff4*b(location4);
                CapacidadeForno(4,t+2) = CapacidadeForno(4,t+2) - diff4;
                DVar4(location4) = DVar4(location4) - diff4;
            end
            if CapacidadeForno(6,t+2)>0 && CVariavel6(t)>0 &&
                swarm((t-1)*num_itens+location6,6,1)~=-1;
```

```
                CapProd6 = CVariavel6(t)/b(location6);
                diff = min(D(location6,aux,1), CapProd6);
                diff6 = min(diff,CapacidadeForno(6,t+2));
                swarm((t-1)*num_itens+location6,6,2) =
                    swarm((t-1)*num_itens+location6,6,2) + diff6;
                CVariavel6(t) = CVariavel6(t) - diff6*b(location6);
                CapacidadeForno(6,t+2) = CapacidadeForno(6,t+2) - diff6;
                DVar6(location6) = DVar6(location6) - diff6;
            end
            if (CVariavel2(t)==0 && CVariavel4(t)==0 &&
                CVariavel6(t)==0) ||
                (CapacidadeForno(2,t+2)==0 &&
                CapacidadeForno(4,t+2)==0 &&
                CapacidadeForno(6,t+2)==0);
                break
            end
            EstProd2(location2) = max(EstProd2) + 1;
            EstProd4(location4) = 0;
            RelacaoDem(location6) = max(RelacaoDem) + 1;
        end
        aux = aux + 1;
    end
end

clear CVariavel2 & CVariavel4 & CVariavel6 & EstProd2 & EstProd4;
clear RelacaoDem & Dem6 & location6 & DVar6 & diff6 & minRelacaoDem;
clear location2 & location4 & DVar2 & DVar4 & diff2 & diff4;
clear maxEstProd2 & maxEstProd4 & minEstProd2 & minEstProd4;
clear CapProd2 & CapProd4 & y2 & y4 & x & aux;
clear CapProd6 & y6 & Dem2 & Dem4;
clear AuxPrep2 & AuxPrep4 & AuxPrep6;
```

O programa principal do Matlab, com a programação do PSO, segue abaixo.

```
clc; clear all; close all;

tempo = 0;
tic;

%Dados
D=xlsread('Olaria1.xls','Plan1','demanda');
s=xlsread('Olaria1.xls','Plan1','custopreparacao');
h=xlsread('Olaria1.xls','Plan1','custoestoque');
sp=xlsread('Olaria1.xls','Plan1','tempopreparo');
b=xlsread('Olaria1.xls','Plan1','tempoproducao');
C=xlsread('Olaria1.xls','Plan1','capacidade');

num_periodos = 8;
num_itens = 35;

Prep = (1:num_periodos*num_itens);
Est = (1:num_periodos*num_itens);
for t = 1:num_periodos
    for i = 1:num_itens
        Prep((t-1)*num_itens+i) = s(i);
        Est((t-1)*num_itens+i) = h(i);
    end
end

%-----------------------------------------------------------------
%Dados PSO

swarm_tam = 6;
inercia = 0.5;
cognitivo = 0.8;
social = 0.6;
alpha = -0.1; %velocidade PSO
beta = 0.1; %velocidade PSO
maxIter = 2000;
delta = 0.1; %mutação
gama = -0.1; %mutação
```

```
num_mutacoes = 30;
CForno = 650000;

%-----------------------------------------------------------------
%Início programa

m = 1:num_periodos*num_itens;
n = 1:swarm_tam;

[swarm] = meshgrid(m,n);

swarm = cat(2,swarm');
swarm(:,:,1) = 0;
swarm(:,:,3) = 0;
%swarm(:,:,1) é DP, swarm(:,:,2) é pbest, swarm(:,:,3) é velocidade

swarm(:,:,1) = 1;

CapacidadeForno(1:swarm_tam,1:num_periodos) = CForno;
IndividuosBackwardOlaria %indivíduos ímpares (1, 3 e 5)
IndividuosForwardOlaria %indivíduos pares (2, 4 e 6)


for i = 1:num_itens*num_periodos
    for t = 1:swarm_tam
        if swarm(i,t,2) == 0
            swarm(i,t,1)= -1;
        end
    end
end

Prep = reshape(Prep,1,[]);
Est = reshape(Est,1,[]);
fitness = (1:swarm_tam);
fitness(:) = 0;

swarm(:,:,3) = swarm(:,:,1);

for i = 1:num_itens*num_periodos
```

```
    for t = 1:swarm_tam
        if swarm(i,t,1)==-1
            swarm(i,t,3)=0;
        end
    end
end

[E] = meshgrid(1:num_itens*num_periodos,1:swarm_tam);
E = cat(2,E');
E(:,:,1)=0;

for individuo = 1:swarm_tam
    E = Estoque(num_itens,num_periodos,swarm,individuo,D,E);
    fitness(individuo) = fitness(individuo) + Prep*swarm(:,individuo,3) + Est*E(:,indivi
end

swarm(:,:,3) = swarm(:,:,1);

gbest(1:num_itens*num_periodos,1:swarm_tam,1) = swarm(:,:,3);

y = zeros(num_itens,num_periodos);
w = zeros(num_itens,num_periodos);
z = zeros(1,num_periodos);

%Encontrar os valores fitness da população inicial.
for p = 1:swarm_tam

    if p==1
        FO = 100000000;
    end
    for t = 1:num_periodos-2
        aux = 0;
        for i = 1:num_itens
            if swarm((t-1)*num_itens+i,p,1)==1
                y(i,t) = 1;
                w(i,t) = 0;
                z(t) = 0;
            end
            if swarm((t-1)*num_itens+i,p,1)==0
```

```
            y(i,t) = 0;
            w(i,t) = 1;
            aux = aux + 1;
        end
        if swarm((t-1)*num_itens+i,p,1)==-1
            y(i,t) = 0;
            w(i,t) = 0;
            aux = aux + 1;
        end

        if aux == num_itens
        z(t) = 1;
        end
    end

end


%Quatro indivíduos da população inicial foram alterados para soluções
% factíveis encontradas pelo Lingo.
if p==1
    y=xlsread('Olaria1-1.xls','Plan1','y');
    w=xlsread('Olaria1-1.xls','Plan1','w');
    z=xlsread('Olaria1-1.xls','Plan1','q');
end
if p==2
    y=xlsread('Olaria1-2.xls','Plan1','y');
    w=xlsread('Olaria1-2.xls','Plan1','w');
    z=xlsread('Olaria1-2.xls','Plan1','q');
end
if p==3
    y=xlsread('Olaria1-3.xls','Plan1','y');
    w=xlsread('Olaria1-3.xls','Plan1','w');
    z=xlsread('Olaria1-3.xls','Plan1','q');
end
if p==4
    y=xlsread('Olaria1-4.xls','Plan1','y');
    w=xlsread('Olaria1-4.xls','Plan1','w');
    z=xlsread('Olaria1-4.xls','Plan1','q');
```

```
    end

    clear aux;

    fitness(p) = 100000000;

    y1 = zeros(1,num_itens*num_periodos);
    w1 = zeros(1,num_itens*num_periodos);

    for t = 1:num_periodos
        for i = 1:num_itens
            y1((i-1)*num_periodos+t) = y(i,t);
            w1((i-1)*num_periodos+t) = w(i,t);
        end
    end

    fid = fopen('Input.txt', 'w+');
    fprintf(fid, '%d ', w1);
    fprintf(fid, '~\n');
    fprintf(fid, '%d ', y1);
    fprintf(fid, '~\n');
    fprintf(fid, '%d ', z);
    fclose(fid);

    comando = 'runlingo Olaria01.ltf';
    [status,cndout]=system(comando);

    infac=size(strfind(cndout,'Error Code'));
    if infac==0
        Fit=load('Output.txt');
        if Fit < fitness(p) || (t*p==1)
            swarm(:,p,3) = swarm(:,p,1);
            fitness(p) = Fit;
            if Fit < FO || (t*p==1)
                for ii = 1:swarm_tam
                    gbest(:,ii,1) = swarm(:,p,3);
                end
                FO = Fit;
            end
```

```
        end
    else
        Fit = 100000000;
    end
end

[minfitness,location] = min(fitness);
for t = 1:swarm_tam
     gbest(:,t,1) = swarm(:,location,3);
end

swarm(:,:,2) = 0;

%--------------------------------------------------------------------------
% MOVIMENTAÇÃO PSO (velocidade será a dimensão 2 do vetor swarm e pbest
%    será a dimensão 3)

cont = 1;
Graf = zeros(maxIter,swarm_tam);


for iter = 1:maxIter

    rand1 = rand(num_itens*num_periodos,swarm_tam,1);
    rand2 = rand(num_itens*num_periodos,swarm_tam,1);

    for t = 1:num_periodos-2
        for i = 1:num_itens
            for j = 1:swarm_tam

                swarm((t-1)*num_itens+i,j,2) =
                    swarm((t-1)*num_itens+i,j,2)*inercia +
                    cognitivo*rand1((t-1)*num_itens+i,j,1).*
                    (swarm((t-1)*num_itens+i,j,3)-
                    swarm((t-1)*num_itens+i,j,1)) +
                    social*rand2((t-1)*num_itens+i,j,1).*
                    (gbest((t-1)*num_itens+i,j,1)-
                    swarm((t-1)*num_itens+i,j,1));
                swarm((t-1)*num_itens+i,j,1) =
```

```
                    swarm((t-1)*num_itens+i,j,1) +
                    swarm((t-1)*num_itens+i,j,2);
                if swarm((t-1)*num_itens+i,j,1) > beta
                    swarm((t-1)*num_itens+i,j,1) = 1;
                elseif swarm((t-1)*num_itens+i,j,1) < 0
                    if swarm((t-1)*num_itens+i,j,1) < alpha;
                        swarm((t-1)*num_itens+i,j,1) = -1;
                    elseif swarm((t-1)*num_itens+i,j,1) > alpha;
                        for k = 1:num_itens
                            if swarm((t-1)*num_itens+k,j,1) == 0
                                swarm((t-1)*num_itens+k,j,1) = 1;
                                break
                            end
                        end
                        swarm((t-1)*num_itens+i,j,1) = 0;
                    else
                        swarm((t-1)*num_itens+i,j,1) = -1;
                    end
                elseif swarm((t-1)*num_itens+i,j,1) >= 0
                    for k = 1:num_itens
                        if swarm((t-1)*num_itens+i,j,1) == 0;
                            swarm((t-1)*num_itens+i,j,1) = 1;
                            break
                        end
                        swarm((t-1)*num_itens+i,j,1) = 0;
                    end
                end
            end
        end
    end

    %1a mutação
    for p = 1:swarm_tam

        item = randi(num_itens,1);
        periodo = randi(num_periodos-2,1);
        aleatorio = rand*2-1;

        if periodo == 1 && D(item,periodo,1)>0
```

```
   swarm((periodo-1)*num_itens+item,p,1)=1;
elseif aleatorio >= delta
    swarm((periodo-1)*num_itens+item,p,1)=1;
elseif aleatorio < gama
    swarm((periodo-1)*num_itens+item,p,1)=-1;
else
    if periodo > 1
        if swarm((periodo-2)*num_itens+item,p,1)==0
            aux = 0;
            for j = 1:num_itens
                if swarm((periodo-2)*num_itens+j,p,1)==0
                    aux = 1;
                    break
                end
            end
            if aux == 0
                for j = 1:num_itens
                    if swarm((periodo-1)*num_itens+j,p,1)==0
                        swarm((periodo-1)*num_itens+j,p,1)=1;
                        break
                    end
                end
                swarm((periodo-1)*num_itens+item,p,1)=0;
                if periodo ~= num_periodos
                    if swarm((periodo)*num_itens+item,p,1)==0
                        swarm((periodo)*num_itens+item,p,1)=1;
                    end
                end
            end
        end
        if swarm((periodo-2)*num_itens+item,p,1)==1
            for j = 1:num_itens
                if swarm((periodo-1)*num_itens+j,p,1)==0
                    swarm((periodo-1)*num_itens+j,p,1)=1;
                end
            end
            swarm((periodo-1)*num_itens+item,p,1)=0;
            if periodo ~= num_periodos
                if swarm((periodo)*num_itens+item,p,1)==0
```

```
                        swarm((periodo)*num_itens+item,p,1)=1;
                    end
                end
            end
        end
    end

    y = zeros(num_itens,num_periodos);
    w = zeros(num_itens,num_periodos);
    z = zeros(1,num_periodos);

    for p = 1:swarm_tam

        for t = 1:num_periodos
            aux = 0;
            for i = 1:num_itens
                if swarm((t-1)*num_itens+i,p,1)==1
                    y(i,t) = 1;
                    w(i,t) = 0;
                    z(t) = 0;
                end
                if swarm((t-1)*num_itens+i,p,1)==0
                    y(i,t) = 0;
                    w(i,t) = 1;
                    aux = aux + 1;
                end
                if swarm((t-1)*num_itens+i,p,1)==-1
                    y(i,t) = 0;
                    w(i,t) = 0;
                    aux = aux + 1;
                end

                if aux == num_itens
                z(t) = 1;
                end
            end

        end
```

```
clear aux;

y1 = zeros(1,num_itens*num_periodos);
w1 = zeros(1,num_itens*num_periodos);

for t = 1:num_periodos
    for i = 1:num_itens
        y1((i-1)*num_periodos+t) = y(i,t);
        w1((i-1)*num_periodos+t) = w(i,t);
    end
end

fid = fopen('Input.txt', 'w+');
fprintf(fid, '%d ', w1);
fprintf(fid, '~\n');
fprintf(fid, '%d ', y1);
fprintf(fid, '~\n');
fprintf(fid, '%d ', z);
fclose(fid);

comando='runlingo Olaria01.ltf';
[status,cndout]=system(comando);

infac=size(strfind(cndout,'Error Code'));
if infac==0
    Fit=load('Output.txt');
    if Fit < fitness(p) || (t*p==1)
        swarm(:,p,3) = swarm(:,p,1);
        fitness(p) = Fit;
        if Fit < FO || (t*p==1)
            for ii = 1:swarm_tam
                gbest(:,ii,1) = swarm(:,p,3);
            end
            FO = Fit;
        end
    end
else
    Fit = 100000000;
```

```
        end

        Graf(iter,p) = fitness(p);

    end

        disp(['Iteração: ' num2str(iter)]);
        disp(['Fitness: ' num2str(FO)]);
        if cont == 10
            disp(['Fitness: ' num2str(FO)]);
            cont = 0;
        end
        toc;
        tempo = tempo + toc;
        cont = cont + 1;

end
```